Universidad Autónoma de Madrid
Escuela politécnica superior

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Trabajo Fin de Grado

AGRUPAMIENTO ESPACIO-TEMPORAL DE SECUENCIAS DE VÍDEO MEDIANTE CARACTERIZACIÓN POR LA RESPUESTA DE REDES CONVOLUCIONALES

Autor: Julio Moreno Blanco
Tutor: Marcos Escudero Viñolo
Ponente: Jesús Bescós Cano

Julio 2020

AGRUPAMIENTO ESPACIO-TEMPORAL DE SECUENCIAS DE VÍDEO MEDIANTE CARACTERIZACIÓN POR LA RESPUESTA DE REDES CONVOLUCIONALES

Autor: Julio Moreno Blanco
Tutor: Marcos Escudero Viñolo
Ponente: Jesús Bescós Cano

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio 2020

Agrupamiento espacio-temporal de secuencias de vídeo mediante caracterización por la respuesta de redes convolucionales

Autor: Julio Moreno Blanco

Tutor: Marcos Escudero Viñolo
Ponente: Jesús Bescós Cano

Video Processing and Understanding Lab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politecnica Superior
Universidad Autonoma de Madrid
Julio 2020

Este trabajo ha sido parcialmente financiado por el Ministerio de Economia, Industria y Competitividad del gobierno de España mediante el proyecto: TEC2017-88169-R MobiNetVideo (2018-2020) Visual Analysis for Practical Deployment of Cooperative Mobile Camera Networks TEC2014-53176-R

Resumen

En este trabajo de fin de grado se establece como objetivo el estudio de la viabilidad de la tarea de agrupamiento espacio-temporal de los cuadros de un vídeo capturado por medio de un vehículo en movimiento o una persona caminando mediante las características visuales extraídas tras el desarrollo de entrenamientos de redes neuronales convolucionales. El objetivo último es el de dividir la trayectoria seguida por el vehículo en grupos, atendiendo a sus cambios de orientación o posición (cambio de manzana), basándose exclusivamente en el análisis visual de las imágenes capturadas por la cámara a bordo.

Para ello se diseña una base de datos usando imágenes del repositorio de Google Street View. Esta base de datos estará compuesta por vídeos capturados por el vehículo mientras recorre trayectorias en diferentes lugares del mundo. Una vez obtenida la base de datos, se examina y eliminan las imágenes que presenten distancias muy próximas con su predecesora, lo que resultará en la obtención de trayectorias consistentes con imágenes que, en los entrenamientos no aporten información redundante.

Creada la base de datos, se carga una red pre-entrenada para la tarea de clasificación de imágenes, GoogleNet. Para ajustar esta red a nuestra base de datos se realizan pruebas, barriendo distintos valores de hiperparámetros para establecer la mejor configuración de la red. Una vez encontrados se realizan varios entrenamientos para la clasificación automática de los vídeos, variando la permeabilidad al aprendizaje de sus capas, a fin de aprovechar parte de las características aprendidas en entrenamientos previos.

Habiendo sido entrenadas las redes, se propone extraer características para cada cuadro del vídeo, mediante truncamiento de la red a diferentes niveles. Con estas características, se calculan distancias entre todos los cuadros de cada vídeo, obteniendo matrices de distancias.

Por último, haciendo uso de las matrices de distancias, se realizan múltiples pruebas mediante el uso de distintos algoritmos de agrupamiento, métricas de evaluación de distancias y criterios para la obtención de agrupaciones. En dichas pruebas se evalúa mediante la ejecución de casos prospectivos, la viabilidad de las agrupaciones obtenidas para cada una de las trayectorias.

Los resultados preliminares sugieren que el método diseñado es capaz de obtener agrupaciones espacio-temporales de los cuadros del vídeo que concuerdan aproximadamente con la trayectoria seguida por el vehículo.

Palabras Clave

Agrupamiento espacio-temporal, características, transferencia de aprendizaje, redes neuronales convolucionales, GoogleNet

Abstract

In this final degree project, the objective is to study the feasibility of spatial-temporal grouping of the frames in a video captured by a moving vehicle or a person walking through the visual characteristics extracted by using a Convolutional Neural Network. The ultimate goal is to divide the following path followed by the vehicle into groups, paying attention to its changes in orientation or position (change of block), exclusively in the visual analysis of the images captured by the on-board camera.

To do this, a database is designed using images from the Google Street View repository. This database is made up of videos captured by the vehicle while recording trajectories in different places around the world. Once we obtain the database, we examine and delete the images that have very close distances with their predecessor, which will result in obtaining consistent trajectories with images that, in training, do not provide redundant information.

Created the database, a pre-trained network is loaded for the image classification task, GoogleNet. To fit this network to our database, tests are performed, sweeping different hyperparameter values to establish the best network configuration. Once these are found, several training sessions are carried out to automatically classify the videos, varying the learning permeability of their layers, in order to use part of the characteristics learned in previous training sessions.

Once the networks are trained, it is proposed to extract characteristics for each frame of the video, by truncation of the network at different levels. With these characteristics, we calculate the distances between all the frames of each video, obtaining distance matrices.

Finally, using distance matrices, multiple tests are performed using different grouping algorithms, distance evaluation metrics and criteria for obtaining groupings. In these tests the viability of the clusters obtained for each of the trajectories is evaluated through the execution of prospective cases.

Preliminary results suggest that the designed method is capable of obtaining spatio-temporal groupings of the video frames that approximately agree with the trajectory followed by the vehicle.

Key words

Space-time clustering, features, transfer learning, off-the-shelf features, CNNs, GoogleNet

Agradecimientos

Me gustaría empezar agradeciendo a mi tutor, Marcos Escudero Viñolo, por haberme dado la oportunidad de realizar este TFG con él, por toda la ayuda, dedicación y facilidades que me ha dado a lo largo de todo el proceso.

A mis padres, Elena y José por absolutamente todo, nada de esto habría sido posible sin ellos.

También me gustaría agradecerles a todas las personas con las que he compartido todos estos años, especialmente a Patri y a Juanan, que desde el minuto uno han estado ahí. Es increíble la familia que se forma por el mero hecho de coincidir en una misma carrera.

Por supuesto agradecerles a Marcos y a Iván por lo inconmensurables que son. Hay muy pocas cosas en la vida que mejoren el hecho de que tus amigos se conviertan en tus hermanos, gracias de todo corazón chicos.

Y como no, debo agradecérselo todo a ella, a Celia, mi salvavidas. Gracias por aguantarme, por todo el cariño, por estar ahí para superar los malos momentos que han venido juntos, por todas las palabras de ánimo, por saber cómo comprenderme en cada momento y sobre todo, gracias por hacerme formar parte de tu vida.

Glosario de acrónimos

- **CNN**: *Convolutional Neuronal Network* (Red Neuronal Convolucional).
- **ANN**: *Artificial Neuronal Network* (Red Neuronal Artificial).
- **SGDM**: *Stochastic Gradient Descent with Momentum* (Descenso de Gradiente Estocástico con Momento).
- **API**: *Application Programming Interface* (Interfaz de Programación de Aplicaciones).
- **MSE**: *Mean Square Error* (Error Cuadrático Medio).
- **DB**: *Davies-Bouldin* (Criterio Davies-Bouldin).
- **SH**: *Silhouette* (Criterio Silhouette).
- **CH**: *Calinski-Harabasz* (Criterio Calinski-Harabasz).

Índice general

Agradecimientos ix

Índice de figuras xvi

Índice de tablas xix

1. Introducción **1**

1.1. Motivación 1

1.2. Objetivos 1

1.3. Organización 2

2. Estado del arte **3**

2.1. Introducción 3

2.2. CNNs 3

2.2.1. Definición 3

2.2.2. Hiperparámetros 5

2.2.3. Arquitecturas seleccionadas: Googlenet 8

2.3. Transferencia de aprendizaje 9

2.3.1. Redes pre-entrenadas 10

2.3.2. Off-the-Shelf Features 10

2.4. Agrupación 10

2.5. Google Street View 14

2.5.1. APIs 15

3. Sistema, diseño y desarrollo **17**

3.1. Materiales 17

3.1.1. Base de datos 17

3.1.2. Anotación manual de las agrupaciones 20

3.2. Métodos 23

3.2.1. Redes entrenadas 23
3.3. Extracción de características 25
3.4. Algoritmos de clustering 26

4. Experimentos **27**
4.1. Descripción de los experimentos 27
4.2. Entrenamiento de las CNN para clasificación de imágenes 27
4.2.1. Ajuste de los hiperparámetros 27
4.2.2. Entrenamiento de las redes para clasificación 28
4.2.3. Extracción de características 30
4.3. Experimento 1: Distancia con imagen siguiente 30
4.3.1. Objetivo 30
4.3.2. Desarrollo y métricas de evaluación 30
4.3.3. Resultados 31
4.3.4. Discusión 33
4.4. Experimento 2: Agrupación 34
4.4.1. Objetivos 34
4.4.2. Desarrollo y métricas de evaluación 34
4.4.3. Resultados 35
4.4.4. Discusión 38

5. Conclusiones **39**
5.1. Conclusiones 39
5.2. Trabajo futuro 40

Bibliografía **43**

Anexos

A. Tablas **45**
A.1. Trayectorias extraídas 45
A.2. Resultados agrupamiento 49
A.2.1. Kmeans 49
A.2.2. Linkage 50
A.3. Resultados completos aciertos 50

B. Imágenes **53**

B.1. Histogramas 53

B.2. Más ejemplos de agrupaciones obtenidas 54

Índice de figuras

2.1. Esquema neurona. Adaptada de [1]. 4

2.2. Distintos tipos de aplicación de máscara en una red convolucional. Extraída de [2]. 5

2.3. Visualización de diferentes curvas para distintos valores para el *learning rate*. Extraída de [3]. 7

2.4. Gráfica relación *accuracy* y número épocas. Extraída de [4]. 7

2.5. Arquitectura de GoogleNet. Extraído de [5]. 8

2.6. Módulo Inception. (a) original. (b) con convolución 1×1. Extraído de [5]. 9

2.7. Comparación en distintas tares usando CNN off-the-shelf features, CNN off-the-shelf features + Data Augmentation y CNN específica. Extraída de [6]. 9

2.8. Ejemplo kmeans. (a) el set original de datos. (b) inicialización aleatoria *clústers*. (c-f) iteraciones del algoritmo hasta hallar la solución. Extraída de [7]. 11

2.9. Ejemplo algoritmo linkage con umbral. Adaptada de [8]. 12

2.10. Cobertura mundial de Google Street View. Extraída de [9]. 15

2.11. Ejemplos de cambios en imágenes. (a) original. (b) heading = 45. (c) fov = 50. (d) pitch = 25. 16

2.12. Ejemplo de ruta. Clase: Parques. 16

3.1. Diagrama de bloques de los pasos a seguir para la extracción de la base de datos. 18

3.2. Ejemplos comandos extracción imágenes. 19

3.3. Cálculo distancias canales RGB de las imágenes. 20

3.4. Distancias imágenes Golden Gate con umbrales. 21

3.5. Ejemplos imágenes redimensionadas de distintas clases. 21

3.6. Ejemplos trayectorias de distintas clases. 22

3.7. Cambio hipotético de *clúster* por giro. Clase: Pueblos mediterráneos. . . 23

3.8. Estimación del número de *clústers* por clase partiendo de la primera hipótesis. Nótese, que sólo se anotan las secuencias de *Test*, una por clase. . . 23

3.9. Comando separación en imágenes de entrenamiento y validación. 24
3.10. Comando activación capa. 26
3.11. Cálculo matriz distancias entre características. La función por defecto calcula la distancia *euclidean*. 26

4.1. Gráfica entrenamiento completo con 103 capas congeladas. 29
4.2. Comparación entre el *vectorA* con los valores obtenidos de la mínima distancia y el *vectorB* el cual es el orden temporal de la secuencia de vídeo. 31
4.3. Histograma del número de aciertos por red y clase. 32
4.4. Histograma mejores capas por clase y red. 32
4.5. Ejemplo distancias Barrios bajos net 41, capa inception_5a-3x3. 33
4.6. Ejemplo distancias Barrios bajos net 41, capa new_classoutput. 33
4.7. Pasos para la extracción del número de *clústers* utilizando la función *evalcluster*. 35
4.8. Representación en formato *boxplot* de la divergencia entre el número de agrupaciones obtenidas y anotadas. 36
4.9. Histograma resultado del *clustering* para todos los casos, más el *ground truth*, con el algoritmo Kmeans. 36
4.10. Trayectoria clase Avenidas extraída de Google Street View y representación de los *clústers* anotados: 5 y obtenidos: 5. 37
4.11. Trayectoria clase Parques extraída de Google Street View y representación de los *clústers* anotados: 5 y obtenidos: 3. 37
4.12. Trayectoria clase Rotondas extraída de Google Street View y representación de los *clústers* anotados: 5 y obtenidos: 5. 37
4.13. Trayectoria clase Medieval extraída de Google Street View y representación de los *clústers* anotados: 5 y obtenidos: 7. 38

B.1. Histograma resultado del *clustering* para todos los casos, más el *ground truth*, con el algoritmo Linkage. 53
B.2. Trayectoria clase Montañoso extraída de Google Street View y representación de los *clústers* anotados: 4 y obtenidos: 15. 54
B.3. Trayectoria clase Plazas extraída de Google Street View y representación de los *clústers* anotados: 4 y obtenidos: 7. 54
B.4. Trayectoria clase Puentes extraída de Google Street View y representación de los *clústers* anotados: 1 y obtenidos: 2. 55

Índice de tablas

3.1. Ejemplo de distintos valores de *fpx* en función de la distancia de la trayectoria. En el Anexo A.1 se muestran todas las trayectorias extraídas con coordenadas de inicio y fin, distancia en km y valores *fpx*. 20
3.2. Acrónimos para las distintas capas congeladas. 25
3.3. Capas seleccionadas para el estudio de las distancias entre características. 26

4.1. Resultado de las pruebas realizadas barriendo hiperparámetros. 28
4.2. Código de colores para los distintos tipos de errores ocurridos durante la ejecución de los entrenamientos. 29
4.3. Resultados entrenamientos. 29
4.4. Acrónimos para las combinaciones en la extracción de características. . . 30
4.5. Porcentajes de los casos de mayor aciertos por clase y red. 31

A.1. Avenidas 45
A.2. Barrios bajos 45
A.3. Barrios judíos 46
A.4. Medieval 46
A.5. Montañoso 46
A.6. Parques 47
A.7. Plazas 47
A.8. Pueblos mediterráneos 47
A.9. Puentes 48
A.10. Rotondas 48
A.11. Resultado kmeans *clustering sqEuclidean* 49
A.12. Resultado kmeans *clustering cosine* 49
A.13. Resultado kmeans *clustering correlation* 49
A.14. Resultado linkage *clustering sqEuclidean* 50
A.15. Resultado linkage *clustering cosine* 50
A.16. Resultado linkage *clustering correlation* 50

A.17.Tabla completa de aciertos por clase, red y capa. 51

1 Introducción

1.1. Motivación

A medida que avanzan y mejoran las tecnologías, aumenta la recogida de datos sobre las personas. Con estos datos recogidos lo que se intenta es conseguir un reconocimiento del comportamiento de las personas, el cómo se mueven, los hábitos que tienen o los desplazamientos que realizan. Ésta práctica se conoce como *life-logging* [10]. Es comúnmente usada para tareas como pueden ser la medicina (control de una vida saludable), vigilancia y seguimiento de ubicación.

Esta última tarea, seguimiento de ubicación, es la más utilizada en *life-logging*, ya que sirve como base para una gran cantidad de aplicaciones.

En lo concerniente a este TFG y siendo esta la motivación del mismo, a partir del concepto de *life-logging*, se pretende obtener automáticamente resúmenes de recorridos de un vehículo circulando o una persona caminando, ambos en entornos urbanos, y obtener representaciones que permitan describir la actividad de desplazamiento de la persona. A estos resúmenes los denominamos agrupaciones espacio-temporales, dado que ofrecen información sobre el tiempo que pasa un vehículo en cada espacio por el que transita. Para obtener estas agrupaciones, haremos uso de características extraídas automáticamente mediante el uso de redes neuronales convolucionales.

1.2. Objetivos

El objetivo global de este TFG es el de explorar la capacidad, mediante la clasificación automática proporcionada por las CNNs, de conseguir agrupaciones espacio-temporales a través de las características extraídas de escenarios del mundo real. Para lograr este cometido se establecen los siguientes objetivos para este TFG:

1. Comprender los conceptos básicos de funcionalidad de las Redes Neuronales Convolucionales y de la extracción de características.
2. Construcción de una base de datos a partir de imágenes capturadas por un vehículo describiendo trayectorias reales alrededor del mundo usando la herramienta Google Street View.
3. Entrenar y ajustar una red neuronal convolucional para la clasificación automática de las trayectorias extraídas.
4. Extraer las características, haciendo uso de las redes entrenadas, sobre los vídeos de la base de datos en función de una anotación subjetiva de su entorno de grabación.
5. Estudiar el potencial del esquema diseñado mediante la realización de casos prospectivos.

1.3. Organización

La organización de este TFG será la siguiente:

- En primer lugar se explicarán los recursos, herramientas y tecnologías aplicadas para la realización de este trabajo en el Capítulo 2.
- Posteriormente, en el Capítulo 3, se explicará el método propuesto: la construcción de la base de datos, el entrenamiento de la red para clasificación, la extracción de las características y los esquemas de agrupamiento evaluados.
- A continuación, en el Capítulo 4, se explicará el desarrollo de los experimentos llevados a cabo, mostrando y discutiendo los resultados en ellos obtenidos.
- Por último en el Capítulo 5 se enunciarán las conclusiones del trabajo y se establecerán algunas líneas de posible trabajo futuro.

2

Estado del arte

2.1. Introducción

En este capítulo explicaremos los diferentes aspectos, materiales y conocimientos en los que se basa este TFG.

Se estudiarán las redes neuronales convolucionales, base del trabajo realizado, los parámetros necesarios para medir y evaluar el entrenamiento de las redes y la transferencia de aprendizaje.

Por último, explicaremos las herramientas necesarias para la extracción de imágenes a partir de Google Street View.

2.2. CNNs

Las Redes Neuronales Artificiales (ANN, por sus siglas en inglés) son en la actualidad ampliamente utilizadas para el análisis, la caracterización, la clasificación y la segmentación de señales. Estas redes se sustentan en la analogía encontrada en la capacidad de observación y aprendizaje desarrollado por el cerebro humano, basándose en su célula más fundamental, la neurona [11].

En particular, para el análisis de imágenes y vídeo, se utiliza principalmente un tipo de ANN: las redes neuronales convolucionales (CNN, por sus siglas en inglés).

2.2.1. Definición

El objetivo del papel que desempeña la analogía presente sobre la neurona es el de transformar todas las posibles entradas en una única salida, pudiendo esta ser a su vez

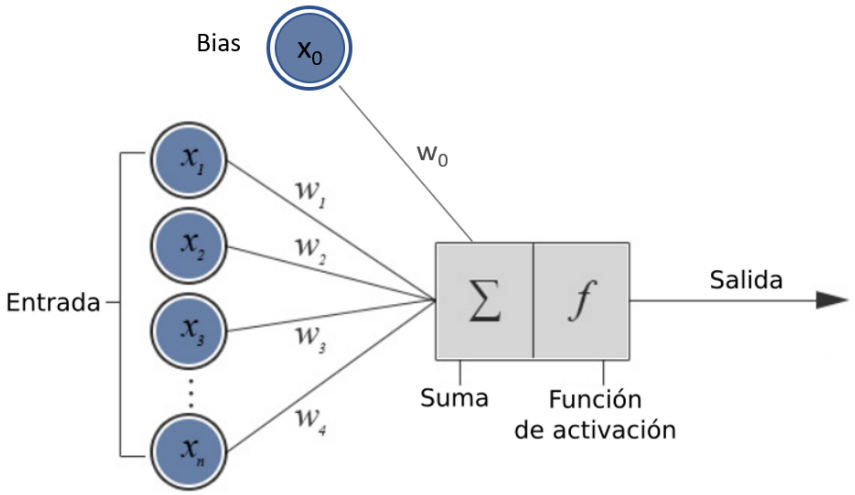


Figura 2.1: Esquema neurona. Adaptada de [1].

entrada de próximas neuronas. Como se muestra en la Figura 2.1, las características que definen el esquema de esta analogía son:

- **Entradas**: formado por el vector de datos x.

- **Pesos**: vector w, inspirados en las conexiones sinápticas de una neurona real. Cada valor de peso es único para cada neurona y se va ajustando a medida que la red neuronal va aprendiendo. Se multiplica por el valor de entrada del vector x correspondiente.

- La **suma** de todas las multiplicaciones de los valores de los pesos con su respectivo valor de entrada, es el valor de entrada a la función de activación f. En ocasiones se suma el valor de Bias, con su correspondiente valor de peso.

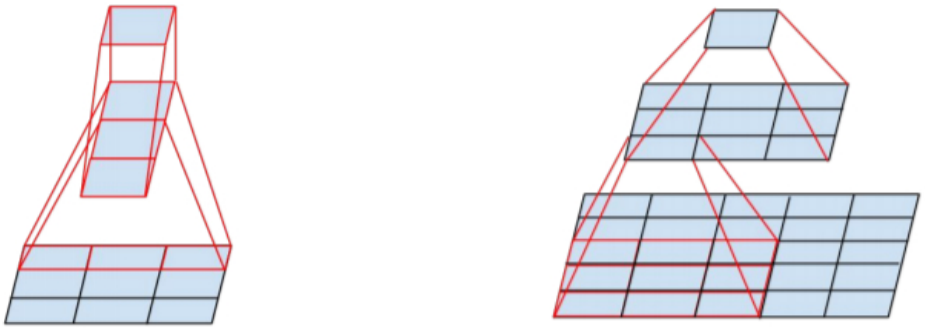
- Esta **función de activación** se encarga de mantener los valores en un rango determinado, normalmente entre $(0, 1)$ o $(-1, 1)$.

Las CNNs están formadas por un número de capas consecutivas, cada una de ellas con una tarea específica. Cabe destacar las siguientes:

- **Capas convolución**: Es en estas capas es donde se produce el filtrado usando una máscara o kernel creando mapas de características. Se puede ver un ejemplo de este tipo de capas en la Figura 2.2. Este filtro se va moviendo por los valores del vector de entrada, obteniendo una matriz a la salida de la capa. Esta matriz de salida será a su vez, matriz de entrada de la siguiente capa de convolución. La estructura de las CNNs se basan en la concatenación de capas de este tipo, intercaladas con capas de diezmado.

- **Capas activación**: En estas capas de la red se escoge la información que avanza a las siguientes etapas de la red. Esta elección se realiza mediante funciones de activación, de las cuales, la más usada, y la que se utiliza en este TFG es la función de activación ReLu. Esta función consiste en coger la información de aquellos valores mayores que cero e igualar los valores negativos a cero, permitiendo el aprendizaje de relaciones no lineales, quedándose sólo con los valores positivos [12].

- **Capas diezmado**: Este tipo de capas están colocadas después de capas convoluciones y/o de activación [13]. La principal funcionalidad de estas capas es la reducción del tamaño de la entrada a capas sucesivas, permite reducir el número de parámetros a entrenar.



(a) Reemplaza convolución 3×3. Convolución 3×1 con 3 salidas.

(b) Reemplaza convolución 5×5. Convolución 3×3 con 9 salidas.

Figura 2.2: Distintos tipos de aplicación de máscara en una red convolucional. Extraída de [2].

2.2.2. Hiperparámetros

Para conseguir un mejor funcionamiento a la hora de utilizar CNNs se deben de ajustar ciertos parámetros conocidos como hiperparámetros. Estos hiperparámetros determinarán la arquitectura de la red así como la forma en la que ésta será entrenada.

Para comprender la influencia que tienen estos parámetros en las CNNs debemos primero entender los principios de tales redes. En primer lugar es necesario saber que el entrenamiento requiere inicializar un primer modelo, el cual se irá ajustando hasta encontrar uno que se adecúe y proporcione unos buenos resultados para la tarea objetivo. La medición de estos resultados viene principalmente dada por la función de pérdida o *Loss* del sistema y del ajuste que proporciona el entrenamiento hacia atrás o *Backpropagation*

Loss

El objetivo es minimizar la *Loss* durante los entrenamientos, asumiendo que los pesos aprendidos puedan servir para datos que no han sido observados durante los entrenamientos. La *Loss* se obtiene comparando la salida de la red con las etiquetas marcadas, conociéndose esta práctica como aprendizaje supervisado.

El ajuste de los pesos se realiza midiendo el impacto de cada uno de ellos en la reducción de la *Loss*, para la realización de esta tarea, se calculan las derivadas parciales de la función *Loss* de cada peso mediante el algorirmo de *Backpropagation* [14].

Hay muchas funciones distintas para poder evaluar el comportamiento de las redes neuronales. Se pueden diferenciar dos principales clases en función de la tarea que se vaya a realizar. Estas clases son: Pérdidas de Regresión (*Regression Loss*) y Pérdidas de Clasificación (*Classification Loss*).

En las clases anteriormente mencionadas, se pueden encontrar diferentes funciones. Dentro de las *Pérdidas de Regresión* se encuentra el Error Cuadrático medio (*Mean Square Error, MSE*), también llamada Pérdida L2 (*L2 Loss*). Esta función es la más usada dentro de las funciones de pérdida de Regresión. En aprendizaje supervisado se basa en el cálculo

del promedio entre la diferencia al cuadrado del valor anotado y el valor pronosticado [13].

Otra función con bastante uso es el Error Absoluto Medio (*Mean Absolute Error, MAE*), también llamada Pérdida L1 (*L1 Loss*), cuya diferenciación respecto de la anterior función es el cálculo en valor absoluto entre el valor anotado y el pronosticado. Por último, aunque menos usada, existe la función Error de Sesgo Medio (*Mean Bias Error, MBE*). Es muy parecida a MAE pero tiene un menor uso debido al no uso del valor absoluto, lo cual puede llevar a la cancelación de los diferentes valores y a no ser una medida congruente [15].

Refiriéndonos ahora al apartado de *Pérdidas de Clasificación*, la función de pérdida más usada para esta tarea, y la utilizada en este TFG, es la Pérdida por Entropía Cruzada *(Cross Entropy Loss)*, más concretamente la Pérdida por Entropía Cruzada Multi Clase *(Multiclass Cross Entropy Loss)*. Se basa en la asignación de un valor para cada una de las clases del set, calculando el promedio de la probabilidad de pertenencia a cada clase.

El resultado se ajusta midiendo la distancia entre el valor anotado y el que se obtiene mediante el uso de la función logaritmo, la cual ofrece como resultado valores pequeños si la distancia es pequeña, y valores grandes si la distancia es grande, lo que permitirá realizar cambios proporcionales para mejorar la clasificación [16].

$$H(y, \widehat{y}) = \sum_i y_i log \frac{1}{\widehat{y_i}} \tag{2.1}$$

$$H(y, \widehat{y}) = \sum_i H(y, \widehat{y}) \tag{2.2}$$

Siendo: y el valor anotado, $\widehat{y}$ el valor pronosticado, y_i valor anotado para la muestra i-ésima, $\widehat{y_i}$ el valor pronosticado categoría i-ésima.

Optimizadores y variables

Una vez vistas las funciones de pérdida se debe explorar qué optimizador aplicamos para el entrenamiento de la red. Los comúnmente más usados son: *Stochastic Gradient Descent with Momentum (SGDM)*, el *Adaptative Moment Optimizer (Adam)* y *Root Mean Squeare Propagation (RMSProp)*. En este TFG se usan SGDM y Adam.

La principal diferencia entre SGDM y Adam es, que SGDM aplica el mismo valor de *learning rate* para todos los pesos (valor α) mientras que el *learning rate* en Adam es adaptativo para los distintos puntos del gradiente. En SGDM se busca la convergencia más rápida dentro de la función de pérdida [17]. Adam solo usa el primer y segundo momento del gradiente (valores de β_1 y $\beta_2 \in [0, 1)$) mientras que SGDM utiliza todos los gradientes anteriores más el actual, lo que hace que tenga una mayor carga computacional [18].

Centrándonos ahora en parámetros más específicos, hay que tener en cuenta los valores que se han de determinar para el resto de hiperparámetros:

- **Mini batch size**: mientras se realiza el entrenamiento, los optimizadores van actualizando los parámetros utilizando una muestra del total de datos. El tamaño de esa

muestra viene dado por el valor que se le asigne a *Mini batch size* [19]. Con tamaños muy grandes, se llevan a cabo un menor número de actualizaciones de los valores.

- **Learning rate**: para valores grandes de *learning rate* es más probable que no se alcance una solución óptima, pues no se llegará al punto de convergencia. Esto se debe a que al dar saltos grandes, pueden saltarse mínimos locales. Sin embargo, valores pequeños de *learning rate* harán que el entrenamiento vaya más lento, pero que se acabe llegando al punto de convergencia, o a un punto mucho más cercano al punto de convergencia, con lo que se tendrá mejores resultados [19]. Se pueden ver ejemplos para distintos valores de *learning rate* en la Figura 2.3.

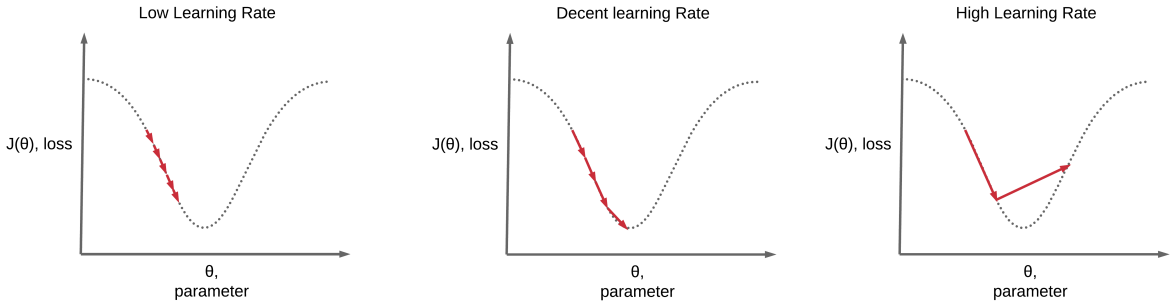


Figura 2.3: Visualización de diferentes curvas para distintos valores para el *learning rate*. Extraída de [3].

- **Número épocas**: una época es un ciclo completo de entrenamiento sobre todo el conjunto de datos destinados para el entrenamiento. A menor número de épocas se puede parar el entrenamiento, de forma que si siguiera, se pudiera llegar a mejores resultados, pero estos no se alcanzarían. Del mismo modo, un número demasiado grande de épocas puede alargar de forma no deseada el entrenamiento, que puede haber alcanzado el máximo valor de precisión en épocas anteriores y por tanto no mejorarán los resultados. El número óptimo de épocas es en el cual la curva de validación o *Test* se estanca, o comienza a decrecer, como se puede ver en la Figura 2.4.

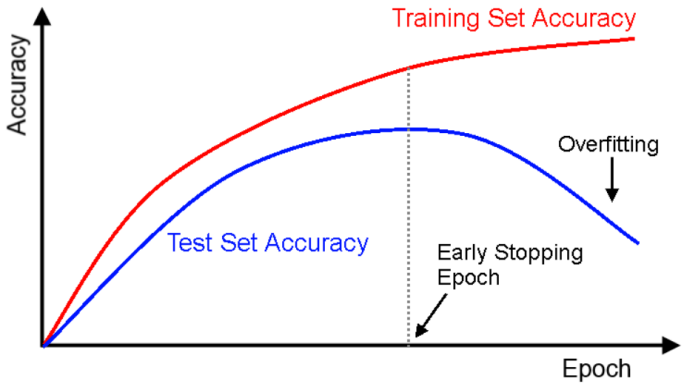


Figura 2.4: Gráfica relación *accuracy* y número épocas. Extraída de [4].

Data Augmentation

En este punto, ya se conoce cómo poder hacer una primera configuración sobre la red. La mayor problemática a la que hay que enfrentarse ahora es la del sobreajuste a los datos (*overfitting*). Para evitar que esto ocurra, la forma más común de corregirlo es mediante *Data Augmentation*.

Su funcionamiento se basa en realizar ciertos cambios aleatorios sobre las datos, los cuales suelen ser rotaciones, traslaciones, escala, desplazamientos, recortes o adición de ruido. Estas alteraciones deben producirse de manera aleatoria, ya que de lo contrario, la red podría adaptarse a esos cambios [20].

2.2.3. Arquitecturas seleccionadas: Googlenet

GoogleNet o *Inception Network* es una arquitectura de red diseñada por Google formada por 22 capas, la cual ha aportado una gran disminución en la tasa de error en comparación con otras redes.

La principal aportación de esta red respecto las demás es el uso de las convoluciones 1×1, las cuales son usadas para disminuir el número de los parámetros de la arquitectura de red y así permitir aumentar la profundidad de la red [5]. Podemos ver la arquitectura de GoogleNet en la Figura 2.5.

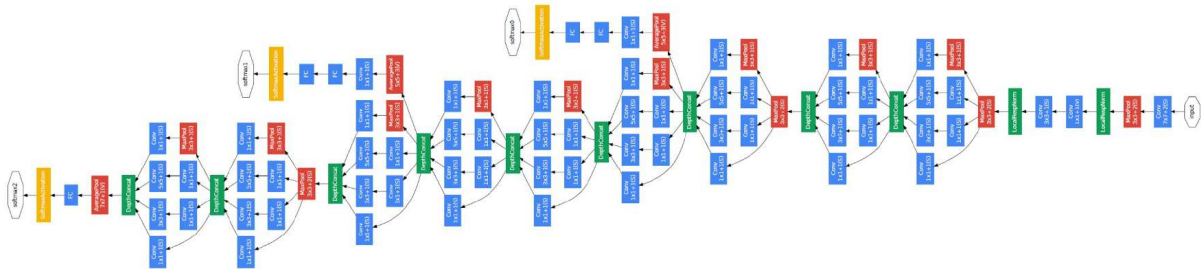


Figura 2.5: Arquitectura de GoogleNet. Extraído de [5].

El módulo clave de la arquitectura es el módulo *inception*: una combinación de las capas convolución 1×1, una convolución 3×3, una convolución 5×5 y una capa de diezmado max-pooling, concatenando los filtros en un único vector de salida. Este vector de salida, será la entrada del siguiente bloque *Inception*. Se puede ver en la Figura 2.6(a) la configuración sin el añadido del uso de las convoluciones 1×1 y en la Figura 2.6(b) con el uso de dichas convoluciones.

Las capas *3 × 3 reduce* y *5 × 5 reduce* representan el uso de la convolución 1×1 antes de las capas convolucionales 3×3 y 5×5 respectivamente.

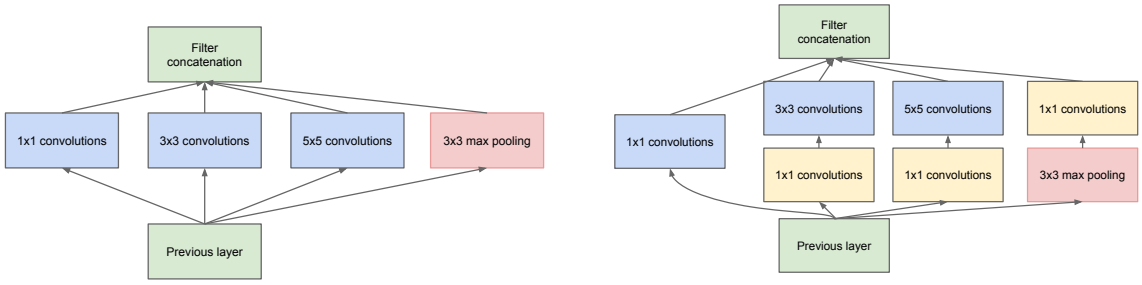


Figura 2.6: Módulo Inception. (a) original. (b) con convolución 1×1. Extraído de [5].

2.3. Transferencia de aprendizaje

La transferencia de aprendizaje se basa en la idea de reutilizar el conocimiento adquirido de modelos previamente entrenados para entrenar nuevos modelos para la misma tarea, en este caso la clasificación automática de imágenes.

La transferencia de aprendizaje se puede dar mediante el ajuste de pesos de una o varias capas de la red, o de extraer descriptores a mayor nivel de profundidad (pesos internos de una capa de red). Estos ajustes se realizan durante el entrenamiento y validación de los nuevos datos de entrada, en este caso imágenes, para la nueva tarea [21].

El uso de transferencia de aprendizaje mejora considerablemente los resultados de la clasificación. Estos resultados son aún mejor si además de la transferencia se le añade Data Augmentation [6].

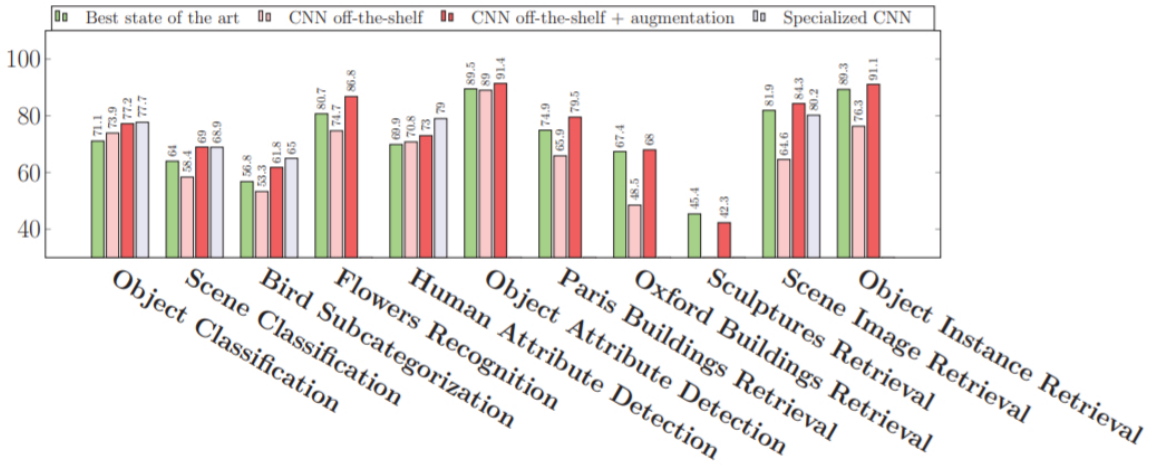


Figura 2.7: Comparación en distintas tares usando CNN off-the-shelf features, CNN off-the-shelf features + Data Augmentation y CNN específica. Extraída de [6].

2.3.1. Redes pre-entrenadas

Aprovechando la característica de transferencia de aprendizaje, se pueden encontrar multitud de redes entrenadas para un mismo problema, lo que permite establecer esa red pre-entrenada como punto de partida a la solución de nuestro problema. Estas redes entrenadas pueden estar orientadas tanto para la tarea de reconocimiento como de clasificación. En lo concerniente a este TFG, la red pre-entrenada que se utilizará estará orientada para la clasificación automática de imágenes.

La utilización de este tipo de sistema ahorra muchos recursos computacionalmente debido a que al estar orientada ya a la tarea requerida, no hay que realizar los ajustes previos para establecer los parámetros de orientación hacia la clasificación.

2.3.2. Off-the-Shelf Features

Teniendo una red pre-entrenada, ésta se puede utilizar como caracterizador para los nuevos modelos de entrenamiento, lo que se conoce como *Off-the-Shelf Features* [6].

Puesto que al haberse realizado un entrenamiento previo, esa red pre-entrenada tiene las características (*features*) listas para el entrenamiento, se pueden ir congelando distinto número de capas a distintos niveles de profundidad de la red. Con esta congelación de capas, solo habría que volver a entrenar el nuevo modelo, un cierto número de capas de la red para que el nuevo modelo, fuera ajustando los pesos de las capas que sí que habría que volver a entrenar, haciendo así que se adaptase a las nuevas imágenes proporcionados.

Esto permite que, al no tener que realizar un entrenamiento de la red entera, computacionalmente sea menos costoso, ya que hay menos capas que entrenar y menos pesos que ajustar.

2.4. Agrupación

La tarea de agrupación consiste en la asociación en grupos o subgrupos de los datos en función de la similitud que presenten con el resto de elementos o grupos.

En relación a este TFG se han utilizado dos algoritmos de clustering: Kmeans [22] y Linkage [23], y cuatro criterios para la selección del número óptimo de grupos: Davies-Bouldin [24], Silhouette [25], Calinski-Harabasz [26] y Gap [27].

Kmeans

Kmeans es un algoritmo de clustering que se basa en la medida de las distancias de los centroides con cada uno de los elementos, seleccionando como partícipe de cada grupo el que resulte en una menor distancia respecto al centroide de ese grupo en comparación con las distancias al resto de centroides [22].

El funcionamiento de Kmeans consiste en:

1- Realizar una primera elección de cuántos grupos k se van a querer tener. Sobre estos k grupos, se parte de una hipótesis cualquiera sobre la inicialización de los valores de los centroides.

2- Una vez inicializados esos valores se calculan las distancias de los centroides elegidos a los datos y se asigna cada uno de los datos a su centroide más cercano, creando k *clústers*.

3- Realizada la asignación a través de la distancia seleccionada, se procede al cálculo de un nuevo centroide para cada *clúster*, repitiéndose los pasos 2 y 3 hasta que en dos iteraciones consecutivas del algoritmo los valores de los k centroides no cambien.

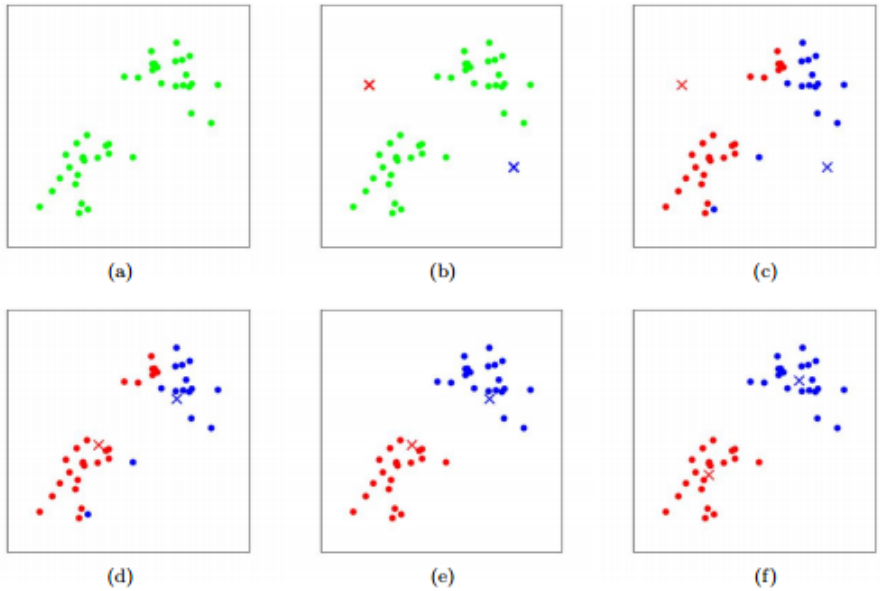


Figura 2.8: Ejemplo kmeans. (a) el set original de datos. (b) inicialización aleatoria *clústers*. (c-f) iteraciones del algoritmo hasta hallar la solución. Extraída de [7].

Linkage

El algoritmo de Linkage se basa en realizar un agrupamiento jerárquico obteniendo un diagrama de árbol o dendograma a medida que se va realizando el algoritmo. Linkage considera en un primer momento a cada una de las observaciones del conjunto de datos como un *clúster* individual, de forma que si existen n datos, habrá n clústers. [23]

El funcionamiento de Linkage es el siguiente:

1- Se parte de un cuadro I_i, el cual es descrito por un vector $\mathbf{f}_i = \mathbf{f}(I_i)$.

2- Se calculan las distancias entre cada par de descriptores: $d(\mathbf{f}_i, \mathbf{f}_j), i, j \in [1, N]$, donde N es el número de cuadros.

3- Una vez calculadas las distancias entre todos los descriptores de todos los cuadros, se guardan los valores en una matriz, (matriz de distancias) y se escoge el valor mínimo de todas las distancias. Se anota el par de cuadros los cuales la distancia entre sus descriptores ha arrojado el valor mínimo, se agrupan, siendo el primer conjunto creado.

4- Se elimina de la matriz de distancias la fila del punto de agrupación seleccionado.

5- Se repiten los puntos 3 y 4 hasta que únicamente quede un único grupo.

Cuando el algoritmo termina y se obtiene el dendograma final, se puede establecer un umbral para que en vez de agrupar todos los datos en un mismo *clúster*, se puedan elegir

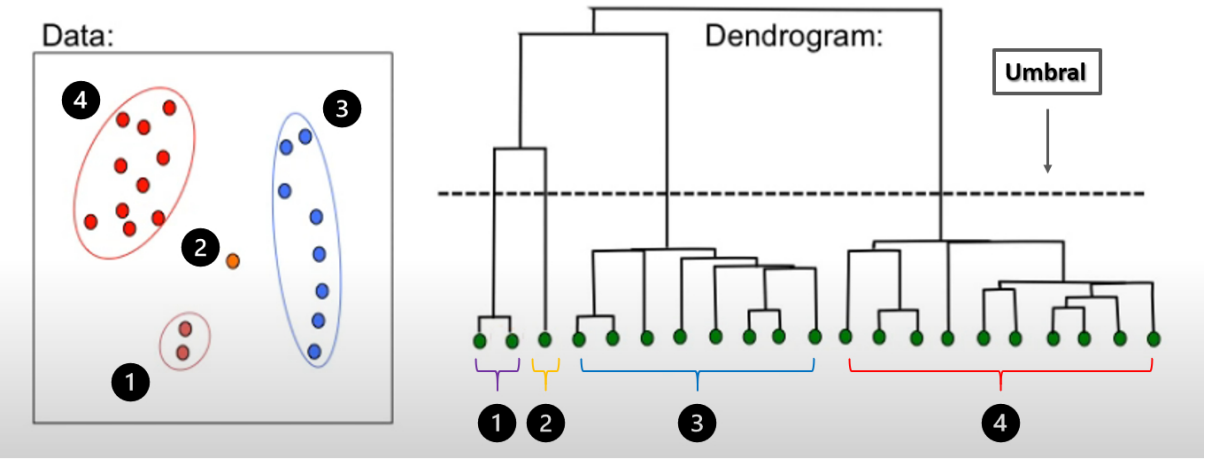


Figura 2.9: Ejemplo algoritmo linkage con umbral. Adaptada de [8].

la cantidad de *clústers* como se aprecia en la Figura 2.9.

Davies-Bouldin

El índice de Davies-Bouldin tiene en consideración las distancias de los datos respecto de los centroides del grupo al que pertenecen como la distancia entre los diferentes clústers que existan [24]. A menor índice de Davies-Bouldin, mejor es la evaluación de los clústers obtenidos.

El índice de Davies-Bouldin se calcula a partir de la siguiente fórmula:

$$indiceDB = \frac{1}{n}\sum_{i=1}^{n} \max_{i \neq j}\left(\frac{\sigma_i + \sigma_j}{d(\mathbf{c}_i, \mathbf{c}_j)}\right) \tag{2.3}$$

Siendo:

n el número de *clústers*

$\mathbf{c}_i$, $\mathbf{c}_j$ los centroides de los *clústers* i-ésimo y j-ésimo

σ_i, σ_j las distancias medias de todos los elementos de los *clústers* i-ésimo y j-ésimo a los centroides i-ésimo y j-ésimo respectivamente

$\mathrm{d}(\mathbf{c}_i, \mathbf{c}_j)$ la distancia euclídea entre los dos centroides de los clústers i-ésimo y j-ésimo

Silhouette

El índice de Silhouette está acotado entre -1 y +1. A mayor índice de Silhouette, mejor es la evaluación de los *clústers* obtenidos. Para la asignación de cada elemento a su *clúster* correspondiente, el índice Silhouette lo que mide es la comparación de la similitud entre el dato a evaluar con los datos del mismo *clúster* en el que ha sido agrupado y los datos de los diferentes *clústers* [25].

El índice de Silhouette se calcula a partir de la siguiente fórmula:

$$indiceS = \frac{1}{n}\sum_{i=1}^{n} \frac{b_i - a_i}{\max\left(a_i, b_i\right)} \tag{2.4}$$

Siendo:

n el número de *clústers*

a_i la distancia media entre el centroide del clúster i-ésimo y todos los demás elementos del mismo clúster

b_i el mínimo valor posible entre los valores de las distancias medias entre el centroide del clúster i-ésimo y todos los demás elementos de cualquiera del resto de clústers

Calinski-Harabasz

Se basa en el cálculo de distancias, usando el criterio de suma de cuadrados, para cada una de las posibles particiones $(n-1)$. A mayor índice de Calinski-Harabasz, mejor es la evaluación de los clústers obtenidos. [26]

El índice de Calinski-Harabasz se calcula a partir de la siguiente fórmula:

$$indiceCH = \frac{SS_B}{SS_W} \cdot \frac{(N-n)}{(n-1)} \tag{2.5}$$

Siendo:

n el número de *clústers*

N el número de muestras

SS_B la varianza general entre *clústers* y SS_W la varianza general dentro del *clúster*.

Las cuales se calculan:

$$SS_B = \sum_{i=1}^{n} N_i \parallel \mathbf{c}_i - \mathbf{m} \parallel^2 \qquad\qquad SS_W = \sum_{i=1}^{n} \sum_{\mathbf{f} \in c_i} \parallel \mathbf{f} - \mathbf{c}_i \parallel^2 \tag{2.6}$$

Siendo:

n el número de grupos

$\mathbf{f}_j$ un punto de datos

N_i el número de muestras en el clúster i-ésimo

$\mathbf{c}_i$ el centroide del clúster i-ésimo

$\mathbf{m}$ la media general de los datos de la muestra

$\parallel \mathbf{c}_i - \mathbf{m} \parallel^2$ la norma L2 (distancia euclidiana) entre los dos vectores

Si los grupos están bien definidos, el valor de SS_B será grande y el de SS_W será pequeño.

Gap

Este criterio se basa en la medición de las desviación observada de la varianza total intra-clúster respecto de los valores esperados. A mayor índice de Gap mejor serán los resultados, ya que esto significa que la estructura obtenida del *clustering* se aleja bastante de una distribución uniforme de los puntos [27].

En primer lugar se realiza *clustering* para los datos observados, variando el número de *clústers*,$(n = 1, 2, \ldots, K)$, y calcular para cada uno de ellos la varianza intra-clúster W_n.

$$W_n = \sum_{i=1}^{K} \frac{1}{2N} D_i \tag{2.7}$$

Donde:

N_i es el número de elementos en cada grupo i-ésimo

D_i es la suma de las distancias por pares para todos los elementos de cada grupo i-ésimo

A continuación, generar B sets de datos de referencia y realizar *clustering* sobre cada uno de los B grupos, variando el número de los K *clústers* anteriormente seleccionados. Calcular el estadístico de Gap mediante:

$$Gap_N(K) = E_N^* log(W_K) - log(W_K) \tag{2.8}$$

Donde: E_N^* denota la expectativa bajo un tamaño de muestra n de la distribución de referencia. Se define generando B copias del conjunto de datos de referencia calculando el promedio $log(W_K^*)$.

Establecer $\mathbf{w}$ y calcular la desviación estándar sd_k. Ajustando el posible error de simulación generado en $E_N^* log(W_K)$ se establece s_K:

$$\mathbf{w} = \frac{1}{B} \sum_{b=1}^{B} log(W_{Kb}^*) \tag{2.9}$$

$$sd_k = \sqrt{\frac{1}{b} \sum_{b=1}^{B} (log(W_{Kb}^*) - \mathbf{w})^2} \qquad s_K = sd_k \cdot \sqrt{1 + \frac{1}{B}} \tag{2.10}$$

Por último, escoger el número de *clústers* como el menor k tal que se cumpla:

$$Gap(K) \geq Gap(K + 1) - s_{K+1} \tag{2.11}$$

Métricas Para cada uno de estos criterios, se pueden usar diferentes tipos de métricas de distancias para la evaluación y cálculo de los conjuntos. Para este TFG se han utilizado las métricas de distancia euclidiana cuadrada (*sqEuclidean*), la distancia coseno (*cosine*) y la distancia correlación (*correlation*).

2.5. Google Street View

Google Street View es una herramienta diseñada por Google para acercar al usuario fotografías de vías urbanas. Estas fotografías son de carácter panorámico, ofreciendo imágenes en 360º. Cuenta con varios tipos de modos de transporte para la captación de

las imágenes, que van desde coches y motos hasta triciclos, todos ellos equipados con cámaras capaces de tomar imágenes 360º.

Desde su creación en 2007, han ido expandiéndose a prácticamente todo el globo abarcando un inmenso número de ubicaciones.

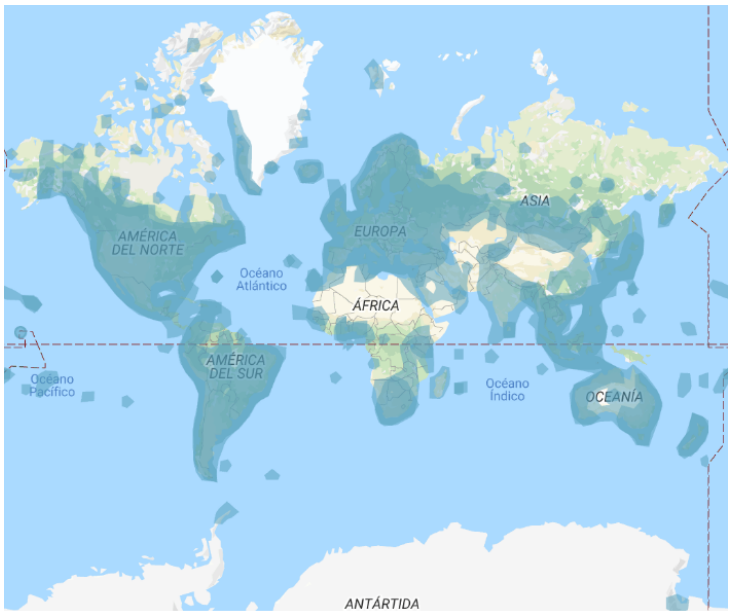


Figura 2.10: Cobertura mundial de Google Street View. Extraída de [9].

Estas fotografías fueron tomadas en diferentes etapas de la herramienta, realizándose desde diferentes tipos de medios de transporte, para poder ofrecer imágenes que siendo obtenidas sólo de una forma, no podrían abarcar todo tipo de escenarios. Los modos usados para este TFG son: modo *walking* y el modo *driving*.

Al hacer uso de esta herramienta nos encontramos con unas variables base que se tendrán que usar para poder acceder a las rutas:

- **Coordenadas**: normalmente en notación de latitud y longitud para establecer las posiciones de inicio y fin.
- **Pitch**: permitirá hacer variaciones en el eje vertical.
- **Heading**: permitirá hacer variaciones en el eje horizontal.
- **Fov**: realiza un efecto de *zoom*.
- **Modo**: el medio de transporte elegido, variando entre andando y conducción.

En la Figura 2.11 se muestran los efectos de estas variables. La imagen original en la Figura 2.11(a) tiene los valores de heading = 0, pitch = 0 y fov = 120.

2.5.1. APIs

En el uso y funcionamiento de Google Street View se pueden encontrar diferentes aplicaciones que consiguen una funcionalidad de la herramienta más completa.

En lo referente a este TFG se han utilizado las siguientes APIs.

Figura 2.11: Ejemplos de cambios en imágenes. (a) original.
(b) heading = 45. (c) fov = 50. (d) pitch = 25.

Street View Publish

Esta API permite subir y publicar imágenes 360 en Google Maps, a cualquier usuario registrado, con datos referidos a ellas, como localización, orientación y conexión con el resto de las imágenes [28].

Directions API

Su funcionamiento se basa en calcular la ruta establecida entre dos puntos del mapa. La trayectoria obtenida es la óptima, en tiempo-distancia, de entre todas las posibilidades, dependiendo del modo que se utilice [29].

Se puede ver un ejemplo de la elección de la ruta óptima en la Figura 2.12. A cada ruta definida se le denominará trayectoria, y estará compuesta de un vídeo grabado desde el vehículo o la persona y de las coordenadas de latitud y longitud de los cuadros que componen el vídeo.

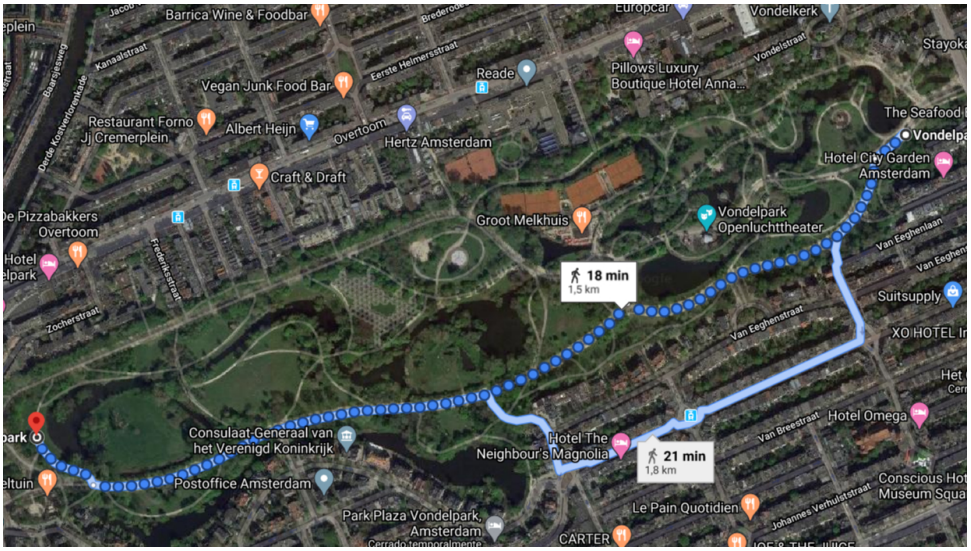


Figura 2.12: Ejemplo de ruta. Clase: Parques.

3

Sistema, diseño y desarrollo

3.1. Materiales

Conociendo las bases del funcionamiento de los elementos utilizados, se explican los materiales que han sido necesarios para el proceder del trabajo.

3.1.1. Base de datos

El diseño de la base de datos se realiza desde cero, pudiendo así organizarla de manera que se adapte a la tarea requerida. Con este diseño propio se busca encontrar una diferencia de trayectorias basada distintos escenarios, tanto urbanos como rurales, pudiendo separarse cada una de las trayectorias y los vídeos que las describen, de forma evidente a la percepción visual.

A cada una de estas separaciones realizadas en base a los distintos tipos de arquitectura las denominaremos clases, habiéndose escogido un total de 10 clases, donde a su vez, cada clase está compuesta por un subconjunto de diez u once trayectorias distintas y sus vídeos asociados. Las clases utilizadas en el desarrollo de este TFG han sido:

- Avenidas, Barrios bajos, Barrios judíos, Medieval, Montañoso, Parques, Pueblos mediterráneos, Plazas, Puentes y Rotondas.

La obtención de estos subconjuntos se diseña para que en cada uno de ellos exista un número de imágenes lo más parejo posible entre sí, con el fin de que no haya descompensación entre clases y por tanto, que determinados de ellos aporten más imágenes y a posteriori puedan aportar más información que el resto de clases en los entrenamientos. El número de imágenes que se quiere extraer para cada trayectoria es de 350 imágenes, aunque este número no será exacto debido a cómo están muestreadas las imágenes en el repositorio de imágenes de Google Street View, y por tanto podrá haber variaciones en el número final de imágenes extraídas por trayectoria.

Extracción base de datos

Para la extracción de las imágenes de la base de datos, se parte de las imágenes pertenecientes a la herramienta Google Street View. Para poder tener acceso a estas imágenes se deben realizar los siguientes pasos, mostrados en la Figura 3.1:

1. Registrarse en Google Cloud Platform.
2. Creación de un nuevo proyecto.
3. En el proyecto creado, solicitar la asignación de una clave API (*API key*), única para nuestro proyecto.
4. Habilitar las APIs *Street View Publish API* y *Directions API*, las cuales junto con nuestra clave API, componen los elementos indispensables para poder hacer la extracción de las imágenes.

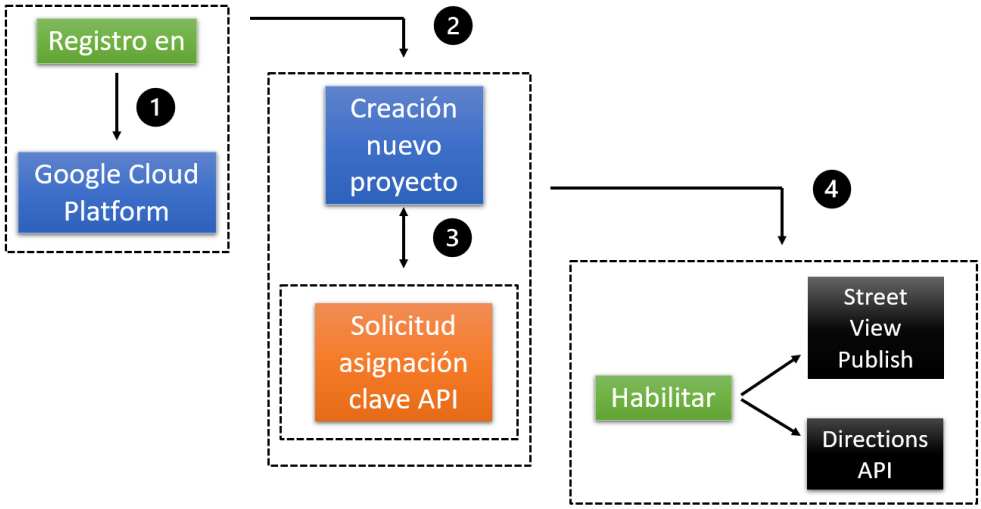


Figura 3.1: Diagrama de bloques de los pasos a seguir para la extracción de la base de datos.

La extracción de imágenes se realiza mediante el uso de un código en lenguaje Java, basado en una extensión de la librería StreetviewExtractor[1], y de Apache Maven para la compilación y ejecución del código. En el código se establece el tamaño de imagen, 600 x 400 píxeles y se fija el valor de *fov* con el que se obtiene mayor rango visual, que se corresponde con un valor de 120. A la hora de ejecutar el código, como se muestra en la Figura 3.2, se pasan por línea de comandos los siguientes argumentos:

- Las coordenadas de origen y destino.

- Se habilita *follow-route*, lo que hará que la orientación de las imágenes apunte en la dirección en la que avanza la trayectoria, definiendo así el pitch y el heading.

- El modo en el que queremos que extraiga las imágenes, *walking* o *driving*.

[1]Link a la librería: https://github.com/jonhare/StreetviewExtractor

- El ajuste de número de imágenes por metro: *fpx*. Este valor se calcula como:

$$fpx = \frac{número\ \ imágenes\ \ requeridas}{distancia\ de\ la\ trayectoria\ (en\ metros)} \tag{3.1}$$

, donde el número de imágenes requeridas es el valor anteriormente mencionado: 350.

- El nombre del archivo de salida. En este caso los nombres de archivo serán los mismos que el de las ubicaciones donde se encuentren las trayectorias.

- Se habilita la extracción tanto del vídeo de la trayectoria como de las imágenes que lo componen.

- Por último se introduce la clave API asignada a nuestro proyecto.

```
java -jar target/StreetviewExtractor-1.0-SNAPSHOT.jar --from -34.609912,-58.381114 --to -34.591857,-58.381979 --follow-route
--mode DRIVING --fpx 0.175 -i -v -o 9deJulioBuenosAires.json -a AIzaSyD3SmbpLTny2RQLIoEM1WBx2FzWN0cMCYo

java -jar target/StreetviewExtractor-1.0-SNAPSHOT.jar --from 40.275201,-5.855646 --to 40.275864,-5.857089 --follow-route --
mode WALKING --fpx 1.16666666666667 -i -v -o HervásCáceres.json -a AIzaSyD3SmbpLTny2RQLIoEM1WBx2FzWN0cMCYo

java -jar target/StreetviewExtractor-1.0-SNAPSHOT.jar --from 55.589880,13.038208 --to 55.577616,13.043171 --follow-route --
mode WALKING --fpx 0.175 -i -v -o RosengårdSuecia.json -a AIzaSyD3SmbpLTny2RQLIoEM1WBx2FzWN0cMCYo
```

Figura 3.2: Ejemplos comandos extracción imágenes.

La organización de estos subconjuntos se realiza en dos carpetas, carpeta *Test* y carpeta *Train + Validation*. Dentro de la carpeta *Test* se coloca una trayectoria de cada clase y todas las trayectorias restantes se colocan en la carpeta *Train + Validation*. Se extraen trayectorias de manera que haya un 10 % de las ellas para la tarea de test y un 90 % de las trayectorias para la tarea de entrenamiento y validación, es decir, se extraen 10 trayectorias por clase.

Problemas y procesamiento de la base de datos

En primera instancia se obtienen un total de 39.525 imágenes. Debido al ajuste de requerir un número parecido de imágenes en cada subconjunto, algunas trayectorias, a causa de tener un menor recorrido en distancia, tienen un valor de *fpx* elevado, lo que provoca que se extraigan muchas imágenes extremadamente cercanas entre si, y por tanto que muchas de las imágenes extraídas presenten escasa o nula diferencia con su predecesora. En la Tabla 3.1 podemos ver ejemplos en los que se muestra cómo varía el valor de *fpx* en función de la distancia de las trayectorias.

Esto provoca que la trayectoria presente inconsistencias, como por ejemplo, que una misma imagen pueda ocupar varias posiciones consecutivas de la trayectoria. Además de este problema, en la extracción de imágenes pueden presentarse otros dos tipos de problemas más:

- *Saltos temporales en las propias imágenes de Google Street View*: puede darse el caso de que en el repositorio de imágenes de la herramienta, ciertos tramos estén almacenados en distintas franjas horarias pasando, por ejemplo, una imagen tomada de día a que su posterior sea de noche, o que una imagen este captada en un día nublado y la siguiente en un día soleado.

Clase	Lugar	Distancia en km	fpx
Avenidas	Las Vegas Blvd S, Las Vegas	2,1	0,17
Pueblos Mediterráneos	Motril, España	0,35	1
Puentes	Puentes del Bósforo, Turquia	0,4	0,875
Rotondas	The Magic Roundabout, Swindon	0,16	2,1875

Tabla 3.1: Ejemplo de distintos valores de *fpx* en función de la distancia de la trayectoria. En el Anexo A.1 se muestran todas las trayectorias extraídas con coordenadas de inicio y fin, distancia en km y valores *fpx*.

- *Saltos posicionales*: en la propia trayectoria, pueden darse cambios abruptos en la situación del medio de transporte durante la trayectoria.

Por lo que, para evitar este tipo de inconsistencias y tener trayectorias más realistas, se lleva a cabo una limpieza de la base de datos. La realización de esta limpieza se realiza calculando la distancia en los tres canales de color (RGB) de cada imagen con su imagen siguiente como se ve en la Figura 3.3, donde *imágenes* es una variable en la cual se van almacenando las imágenes de cada subconjunto de cada clase para poder leerlas y calcular las distancias.

```
for j = 1:numel(imagenes)-1
    distan_imas(j) = sqrt(sum(sum(sum((double(imagenes{j,1}) - double(imagenes{j+1,1})).^2))));
end
```

Figura 3.3: Cálculo distancias canales RGB de las imágenes.

Una vez calculadas las distancias para cada subconjunto, se calcula cuál es el valor máximo de distancia y se establecen dos umbrales, como se ve en la Figura 3.4.

El umbral superior y el inferior se calculan respectivamente como:

$$umbral\_superior = 95\,\% \cdot máxima\ distancia$$
$$umbral\_inferior = 35\,\% \cdot máxima\ distancia$$

Tras realizar dicha limpieza, el número de imágenes total obtenido es de 29.937, un 60,56 % del total de las imágenes. Dicho número de imágenes se redimensionan para cumplir el tamaño de entrada de imagen requerido por la red, que es de $224 \times 224 \times 3$. Se pueden ver ejemplos de las imágenes extraídas y ya redimensionadas en la Figura 3.5.

3.1.2. Anotación manual de las agrupaciones

Se propone un esquema de anotación manual de las trayectorias, el cual se basará en la observación de las trayectorias de la carpeta *Test*. En la Figura 3.6 vemos dos ejemplos de estas trayectorias.

Las características visuales que usaremos para establecer la aparición de un nuevo *clúster* en las trayectorias serán: giros marcados en las trayectorias y cambios de manzana, las cuales se podrán ver en el Capítulo 4. Se puede ver un ejemplo visual de un giro en la Figura 3.7.

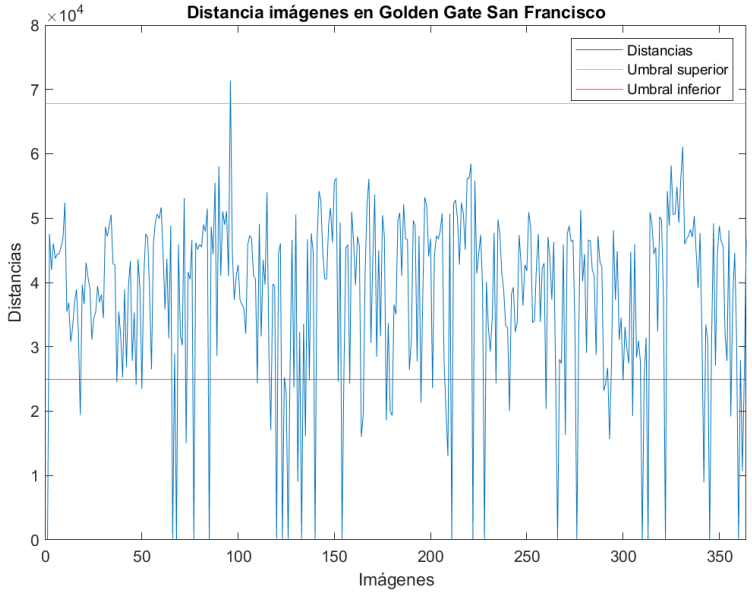
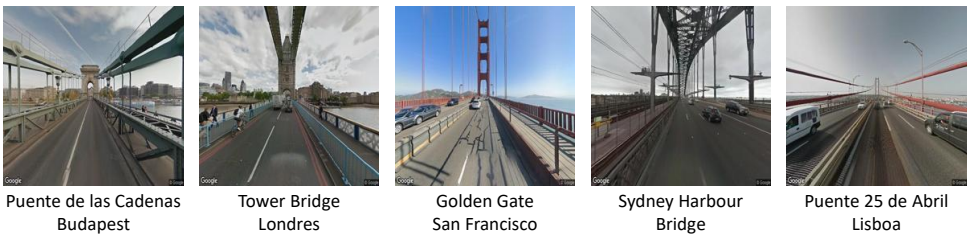


Figura 3.4: Distancias imágenes Golden Gate con umbrales.

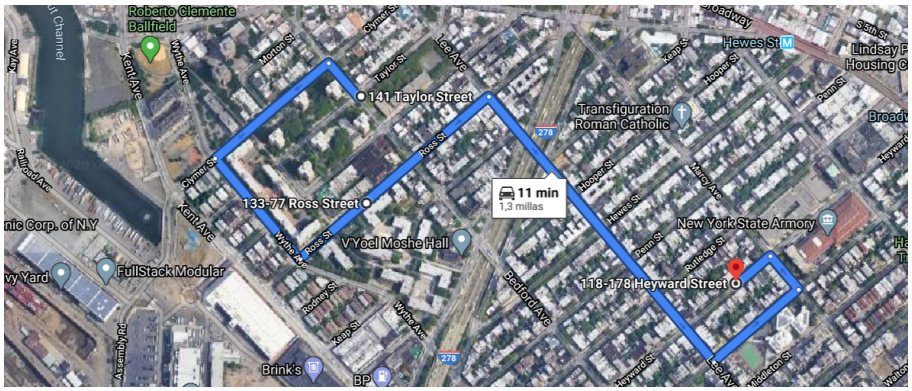
(a) Ejemplos imágenes. Clase: Avenidas.



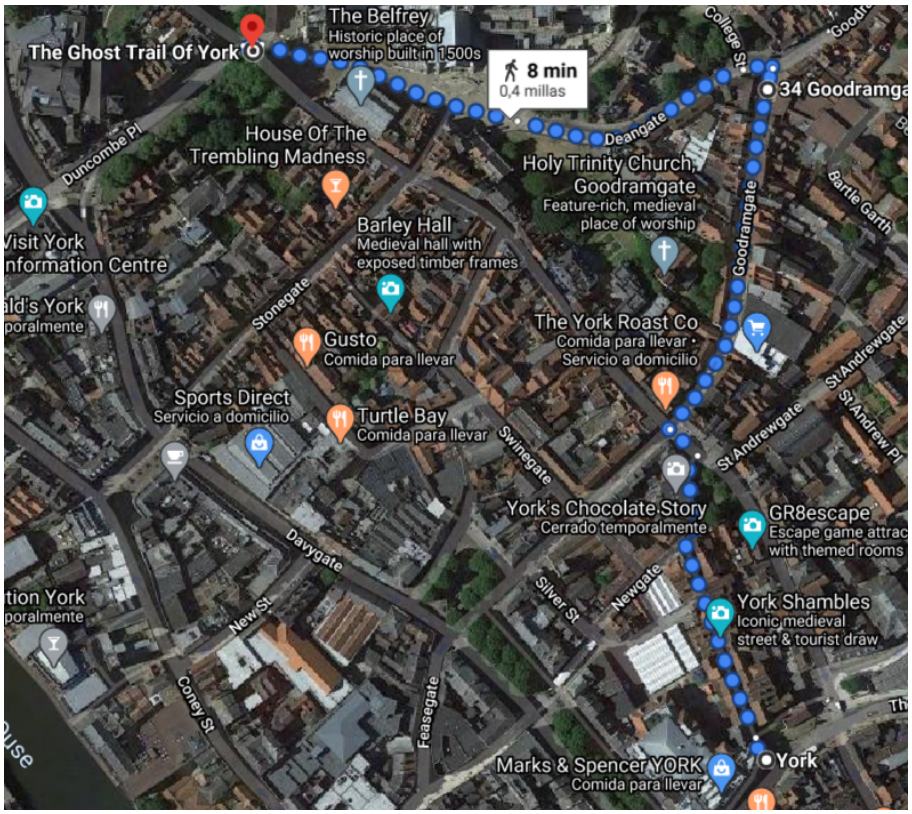
(b) Ejemplos imágenes. Clase: Puentes.

(c) Ejemplos imágenes. Clase: Parques.

Figura 3.5: Ejemplos imágenes redimensionadas de distintas clases.



(a) Trayectoria clase Barrios bajos.



(b) Trayectoria clase Medieval.

Figura 3.6: Ejemplos trayectorias de distintas clases.

Esta primera hipótesis sobre el número de *clústers* esperados, nos servirá para ver el comportamiento y la relación que exista entre los resultados obtenidos una vez hecho el agrupamiento automático a través del método que se propone y las anotaciones manuales realizadas sobre las trayectorias. El número de *clústers* resultantes realizando estas anotaciones manuales para cada clase pueden verse en la Figura 3.8.

Figura 3.7: Cambio hipotético de *clúster* por giro. Clase: Pueblos mediterráneos.

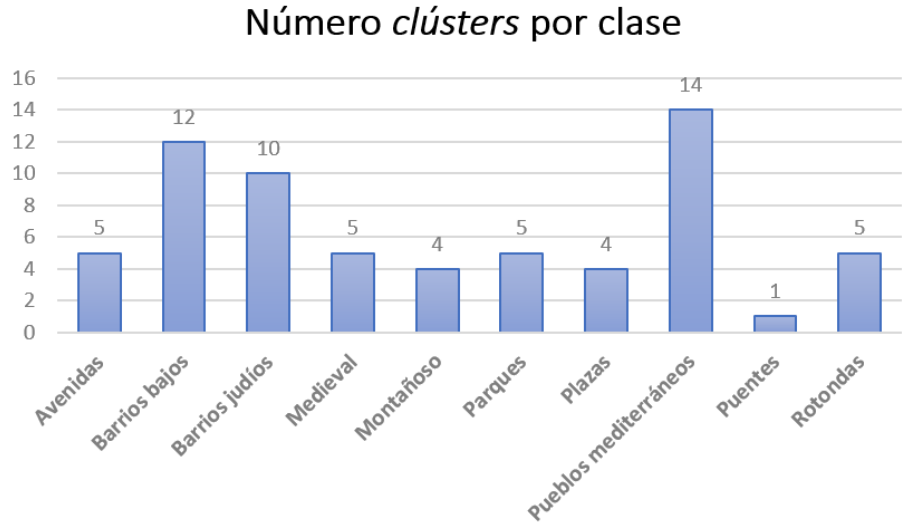


Figura 3.8: Estimación del número de *clústers* por clase partiendo de la primera hipótesis. Nótese, que sólo se anotan las secuencias de *Test*, una por clase.

3.2. Métodos

En este apartado explicaremos cómo se realiza el entrenamiento de la CNN que posteriormente usaremos para la extracción de características, así como las pruebas y decisiones a tomar para conseguir unos mejores resultados.

3.2.1. Redes entrenadas

En primer lugar se separan de forma aleatoria las imágenes contenidas en la carpeta *Train + Validation* mediante la función *splitEachLabel* en Matlab, como se ve en la Figura 3.9 dedicando un 60 % de las imágenes para entrenamiento y un 40 % para validación.

Una vez realizada esta separación de imágenes, para comenzar con los entrenamientos

```
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.6,'randomized');
```

Figura 3.9: Comando separación en imágenes de entrenamiento y validación.

lo primero que se hace es cargar en Matlab la red pre-entrenada para la clasificación de imágenes escogida en este TFG, Googlenet.

Una vez cargada la red, se reemplazan las últimas capas de esta. La última capa se reemplaza para establecer que el número de salidas sea igual al número de clases definidas. La capa de clasificación se reemplaza para eliminar las etiquetas de la red pre-entrenada. Cuando se realicen los entrenamientos, la función *trainNetwork* será quien asigne automáticamente las nuevas etiquetas de salida.

A continuación, una vez que se realiza esta sustitución de las capas finales, se diseñan una serie de cambios, *Data Augmentation*, que se realizan sobre todas las imágenes que se vayan a utilizar. Estos cambios se llevan a cabo como medida preventiva extra para evitar que se produzca *overfitting*. Las alteraciones introducidas en las imágenes en este *Data Augmentation* son:

- Reflexión tanto en el eje x como en el eje y.

- Rotación en un rango de [-120º, 120º].

- Escalado en un rango [0.7, 1.3].

- Traslación en un rango [-60, 60].

Antes de realizar un entrenamiento completo haciendo uso de la totalidad de las imágenes de la base de datos, es necesario llevar a cabo una serie de pruebas para establecer la mejor configuración de la red.

Búsqueda de hiperparámetros

Estas diferentes pruebas se realizan usando una submuestra de la base de datos. En este caso se selecciona un 10 % de imágenes aleatoriamente de cada clase (conjunto de ajuste).

Con objetivo de lograr los mejores resultados posibles, se realizan múltiples combinaciones con los diferentes valores asignados a los hiperparámetros. En las pruebas realizadas los hiperparámetros que se someten a valoración son:

- Optimizador utilizado, variando entre SGDM y Adam.
- *Learning Rate*, variando los valores entre 10^{-3} y 10^{-4}.
- *Mini Batch Size*, variando los valores entre 100, 150 y 200.

El número de épocas se fija en un valor de 1000 tanto para los casos de prueba en la búsqueda de hiperparámetros como para el entrenamiento de la red completa. En el Capítulo 4 se mostrarán los resultados de las pruebas más significativas de estos experimentos.

Configuración de la red

Con el fin de que las imágenes seleccionadas en cada época del entrenamiento no sean siempre las mismas, se establece para la opción *Shuffle* de la red el valor *every-epoch*. Con esto, se tendrá que antes de cada época y cada validación, se barajen las imágenes obteniendo así una aleatorización en el orden en el que éstas pasan por la red.

Por otro lado se dispone una configuración de cláusula de escape. En esta cláusula se establece que si en 250 épocas anteriores a la actual, el valor de precisión de validación (*validation accuracy*) no ha mejorado, el entrenamiento se detiene.

Por último, para evitar la pérdida de resultados debida a cualquier error que se pudiera producir durante el entrenamiento, se habilita la opción de guardado automático. Se establece que este guardado automático se produzca tras un número de iteraciones que coincida con el fin de cada época de entrenamiento. Este número se obtiene de:

$$frecuencia\ guardado = \frac{número\ de\ imágenes\ para\ entrenar}{valor\ Mini\ Batch\ Size} \tag{3.2}$$

Entrenamiento completo

Una vez determinada la configuración de la red, se entrenará la red con todo el conjunto de entrenamiento. Se definirán varios esquemas de entrenamiento, donde se conserven parcialmente los pesos de la red pre-entrenada. Para ello, se añaden pruebas de congelar distintos números de capas de red. Con este congelamiento se busca conseguir dos cosas: añadir elementos que no favorezcan el *overfitting* y aumentar la rapidez de los entrenamientos. Las opciones elegidas en el número de capas a congelar son: 10, 41, 73, 103 y 134. La congelación se realiza de forma incremental, es decir cuando se congelan 10 capas son las 10 primeras empezando desde la entrada, después las 41 primeras, y así sucesivamente.

Los resultados de este estudio pueden consultarse en la sección 4.2. El estudio resulta en la definición de varias redes con distintas precisiones globales en la predicción de la clase dado el conjunto de ajuste. En la sección experimental se utilizarán para referirse a estas redes los acrónimos de la Tabla 3.2.

Acrónimo	net 10	net 41	net 73	net 103	net 134
Capas congeladas	10	41	73	103	134

Tabla 3.2: Acrónimos para las distintas capas congeladas.

3.3. Extracción de características

Una vez seleccionada y entrenadas las redes, para poder empezar a evaluar resultados, se necesita extraer las características de las capas de la red que se quieran estudiar. Esta activación de las capas se realiza mediante el uso de la función *activation* de Matlab como se ve en la Figura 3.10.

```
features = activations(net,imds,layer,'OutputAs','rows');
```

Figura 3.10: Comando activación capa.

Donde: *net* es una de las redes entrenadas, *imds* son las imágenes a estudio y *layer* la capa de red la cual queramos observar. Esta instrucción permite ver en una matriz las características asociadas a cada imagen de la clase y capa seleccionadas. Para evaluar estas características, realizamos el cálculo de la distancia de todas las características extraídas, mediante la función *pdist2* de Matlab, como se ve en la Figura 3.11, obteniendo de esta manera una matriz de distancias entre vectores de características. Este cálculo se usa en el experimento 1 (ver sección 4.3). En el experimento 2 (ver sección 4.4) se utilizarán distintas métricas de distancia para evaluar las distancias entre las características.

```
distancias_entre_features = pdist2(features,features);
```

Figura 3.11: Cálculo matriz distancias entre características. La función por defecto calcula la distancia *euclidean*.

Capa	new classoutput	new prob	new fully	new drop	inception 5b 5x5 reduce	inception 5a 3x3	inception 4c 5x5	inception 4a 3x3 reduce
Número capa	144	143	142	141	132	116	77	43

Tabla 3.3: Capas seleccionadas para el estudio de las distancias entre características.

En este TFG, estudiaremos características extraídas a la salida de diferentes capas, en particular, las capas seleccionadas para la caracterización se pueden ver en la Tabla 3.3. Los resultados de este estudio pueden consultarse en la sección 4.3.

3.4. Algoritmos de clustering

A la hora de realizar el *clustering*, se procede a la comparación entre dos criterios que nos darán, para cada trayectoria seleccionada de la carpeta de *Test*, un número de *clústers* distintos. Las combinaciones de estos criterios para establecer un número de casos de estudio son los siguietens: Davies-Bouldin y Silhouette, Davies-Bouldin y Calinski-Harabasz, Davies-Bouldin y Gap, Calinski-Harabasz y Silhouette, Calinski-Harabasz y Gap y Silhouette y Gap.

Cada uno de estos casos se realizan usando tanto el algoritmo Kmeans como el algoritmo Linkage. A su vez, para cada algoritmo se usan las tres métricas de distancias seleccionadas para este TFG: *sqEuclidean*, *cosine* y *correlation*.

Con estas combinaciones obtendremos suficientes escenarios para la comparación sobre la veracidad de los *clústers* formados así como la discusión sobre los resultados que se verán en la sección 4.4.

4
Experimentos

4.1. Descripción de los experimentos

En esta sección explicaremos cómo después de completar los entrenamientos efectuados, mediante las características obtenidas de dichos entrenamientos, podemos evaluar la agrupación y coherencia de los *clústers* obtenidos para cada uno de los escenarios seleccionados.

Para ello se realizan dos experimentos:

- El primero de ellos se centra en la observación de las distancias entre las características seleccionadas para diferentes capas de red, midiendo si haciendo uso de estas distancias se puede corroborar que la imagen siguiente a la actual tiene menor distancia respecto a su predecesora que el resto de imágenes.
- El segundo experimento, en el cual se centra este TFG, se basa en el estudio de diferentes algoritmos y criterios de evaluación para la generación de distintos *clústers* y en su comparación con la hipótesis formulada de los *clústers* que se suponen se obtendrán, mostrados en la Figura 3.8.

4.2. Entrenamiento de las CNN para clasificación de imágenes

4.2.1. Ajuste de los hiperparámetros

Una vez realizadas las pruebas descritas en la sección 3.2, para establecer los hiperparámetros que se usan en los entrenamientos, así como la configuración de la red, se obtienen los resultados mostrados en la Tabla 4.1. En estas pruebas para ajustar los hiperparámetros, se congelan las 10 primeras capas de la red.

Tras realizar las pruebas, se determina que hay una similitud de resultados entre las dos opciones que devuelven un mayor rendimiento:

- SGDM con un *mini batch size* de 100 y un *learning rate* de 10^{-3}.

- Adam con un *mini batch size* de 150 y un *learning rate* de 10^{-4}.

Como se ve en la Tabla 4.1, la gran diferencia entre estas dos opciones viene dada por la duración del entrenamiento, por lo que se decide escoger la primera de las dos configuraciones mencionadas.

Optimizador	*Mini batch size*	*Learning rate*	*Validation accuracy*	Tiempo	Barajar *(Shuffle)*
SGDM	150	10^{-3}	72.76 %	149' 13"	every-epoch
SGDM	150	10^{-4}	63.07 %	259' 15"	every-epoch
SGDM	200	10^{-3}	79.16 %	284' 56"	every-epoch
SGDM	100	10^{-3}	**80.07 %**	208' 11"	every-epoch
SGDM	100	10^{-4}	76.23 %	285' 33"	every-epoch
Adam	150	10^{-4}	79.71 %	433' 31"	every-epoch
Adam	200	10^{-3}	77.33 %	501' 30"	every-epoch

Tabla 4.1: Resultado de las pruebas realizadas barriendo hiperparámetros.

Por lo tanto, la arquitectura final de la red vendrá dada por la siguiente configuración:

- Optimizador: SGDM.
- Número de épocas: 1000.
- *Mini Batch Size*: 100.
- Barajado (*Shuffle*) de imágenes en cada época.
- *Learning Rate* de 10^{-3}.

4.2.2. Entrenamiento de las redes para clasificación

Una vez determinados los hiperparámetros, se procede al entrenamiento de la red para clasificación de imágenes utilizando el dataset completo. Se realizan varios entrenamientos variando el número de capas congeladas mencionadas en la sección 3.2 del Capítulo 3, obteniendo cinco redes entrenadas de manera diferente. El objetivo es observar cómo varía el rendimiento final de la red si se conserva parte del entrenamiento anterior. Recordemos que la red original precargada estaba entrenada para clasificación de imágenes, pero utilizando otro dataset distinto. Durante estos entrenamientos se suceden varios tipos de errores mostrados en la Tabla 4.2

Debido a estos errores, para cada entrenamiento realizado se hacen dos selecciones. Para todos ellos se selecciona la mejor época, es decir, la época que tenga un valor de precisión en validación más alta así como la última época del entrenamiento.

Para todos los entrenamientos, se mantienen los valores establecidos para la arquitectura de la red final establecidos en la subsección, por lo que no se mostrarán en la Tabla 4.3.

Los resultados obtenidos para cada uno de los entrenamientos son los mostrados en la Tabla 4.3:

Tabla de errores	
Error de caída de la precisión (*Validation Accuracy*) (11,75 %-13,53 %)	
Error de *CUDA* (controlador tarjeta gráfica)	
Sin errores	

Tabla 4.2: Código de colores para los distintos tipos de errores ocurridos durante la ejecución de los entrenamientos.

Acrónimo	Capas congeladas	Capa	*Validation accuracy*	Tipo error
net 10	10	Final	92,95 %	
net 10	10	Mejor	**94,59 %**	
net 41	41	Final	11,75 %	
net 41	41	Mejor	91,62 %	
net 73	73	Final	13,52 %	
net 73	73	Mejor	91,88 %	
net 103	103	Final	82,67 %	
net 103	103	Mejor	83,21 %	
net 134	134	Final	67,87 %	
net 134	134	Mejor	69,93 %	
net 134-2	134	Final	13,52 %	
net 134-2	134	Mejor	71,33 %	

Tabla 4.3: Resultados entrenamientos.

Como se puede observar en la Tabla 4.3, al ir congelando las capas, las tasas de aprendizaje de las capas congeladas no actualizan sus valores, por lo que al tener menos capas para poder entrenar la red, resulta en menores valores de precisión en validación (*validation accuracy*). Se puede ver un ejemplo de gráfica resultante de un entrenamiento en la Figura 4.1. En la evaluación de los experimentos, para la net 134, se tendrán en cuenta los resultados del segundo entrenamiento (net 134-2), siendo una extensión del primero (net 134).

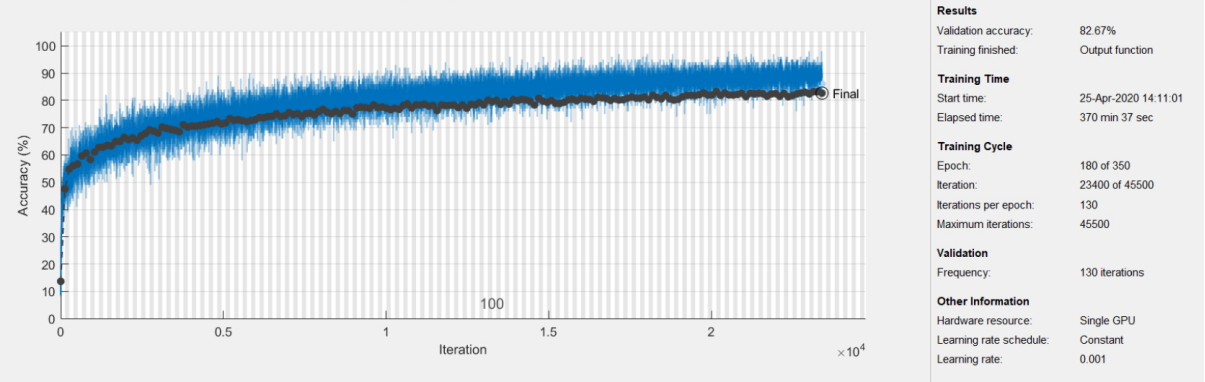


Figura 4.1: Gráfica entrenamiento completo con 103 capas congeladas.

4.2.3. Extracción de características

Para la generación automática de agrupaciones, se usan las trayectorias contenidas en la carpeta *Test*.

Como se explicó en la sección 3.2.1, se evaluará el uso de las cinco redes entrenadas (ver Tabla 4.3) así como de características extraídas a diferentes profundidades o capas de cada una de estas redes. Cada una de estas opciones para la extracción de características se identificará con el acrónimo indicado en la Tabla 4.4. En total se evaluarán 40 extractores de características: 8 tipos de características (correspondientes al truncamiento de 8 capas diferentes) extraídas de 5 redes entrenadas con diferentes congelaciones.

Capa \ Red	net 10	net 41	net 73	net 103	net 134
new classoutput	10 out	41 out	73 out	103 out	134 out
new prob	10 prob	41 prob	73 prob	103 prob	134 prob
new fully	10 full	41 full	73 full	103 full	134 full
new dropout	10 drop	41 drop	73 drop	103 drop	134 drop
inception_5b-5x5 reduce	10 5b	41 5b	73 5b	103 5b	134 5b
inception 5a-3x3	10 5a	41 5a	73 5a	103 5a	134 5a
inception 4c-5x5	10 4c	41 4c	73 4c	103 4c	134 4c
inception 4a-3x3 reduce	10 4a	41 4a	73 4a	103 4a	134 4a

Tabla 4.4: Acrónimos para las combinaciones en la extracción de características.

4.3. Experimento 1: Distancia con imagen siguiente

4.3.1. Objetivo

Como primer experimento se intenta relacionar el posicionamiento de las imágenes de los vídeos con las características de las mismas, es decir, comprobar si a través de las características extraídas de las diferentes capas seleccionadas para estudio, la distancia mínima entre la imagen n y el resto de imágenes se corresponde con la imagen siguiente $(n+1)$. Esto es, si el orden resultante de calcular la distancia mínima se corresponde con el orden temporal de los cuadros del vídeo que describe la trayectoria.

4.3.2. Desarrollo y métricas de evaluación

1- Tomamos como imagen origen la primera imagen del vídeo y, sobre la matriz de las características extraídas, calculamos las distancias de la imagen origen con el resto de imágenes obteniendo la matriz distancia entre características.

2- Seleccionamos la distancia mínima y vemos en qué posición se encuentra, guardando esta posición de la distancia mínima encontrada en un $vectorA$, para su posterior comparación con el orden temporal de las imágenes, guardado en un $vectorB$.

3- Se repiten todos los pasos, cogiendo la imagen siguiente, respecto de la cogida la anterior iteración como imagen origen.

4- Una vez se llega a la última imagen de la trayectoria bajo análisis, se comparan ambos vectores, como se muestra en la Figura 4.2.

5- De esta comparación se contarán las veces que los valores entre ambos vectores coinciden, con lo que cada coincidencia supondrá un acierto.

$$acierto_i = \begin{cases} 1 & \text{si } a_i - b_i = 0 \\ 0 & \text{si } a_i - b_i \neq 0 \end{cases} \qquad \text{donde } i = 1, 2, 3, \ldots, (N-1), N. \tag{4.1}$$

Donde N es el número total de imágenes del vídeo, a_i y b_j son las posiciones i-ésimas de cada vector y $acierto_i$ es el acierto en la posición i-ésima.

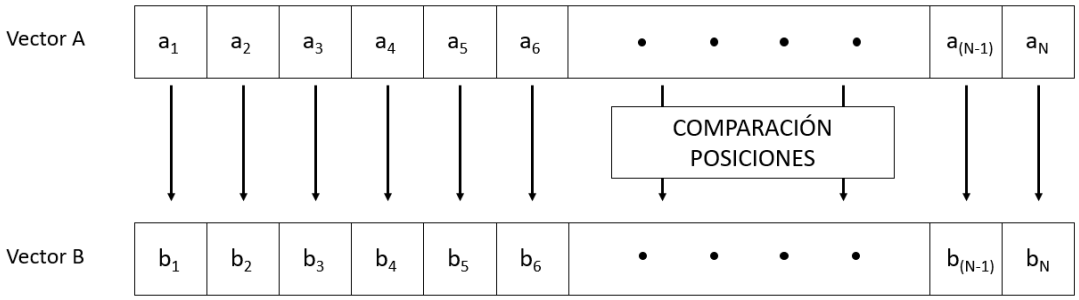


Figura 4.2: Comparación entre el $vectorA$ con los valores obtenidos de la mínima distancia y el $vectorB$ el cual es el orden temporal de la secuencia de vídeo.

6- Por último, calculamos el porcentaje de aciertos, sumando el total de los aciertos y dividiendo entre el número de imágenes de la trayectoria N.

$$porcentaje\ aciertos = \frac{1}{N} \sum_{i=1}^{N} acierto_i \tag{4.2}$$

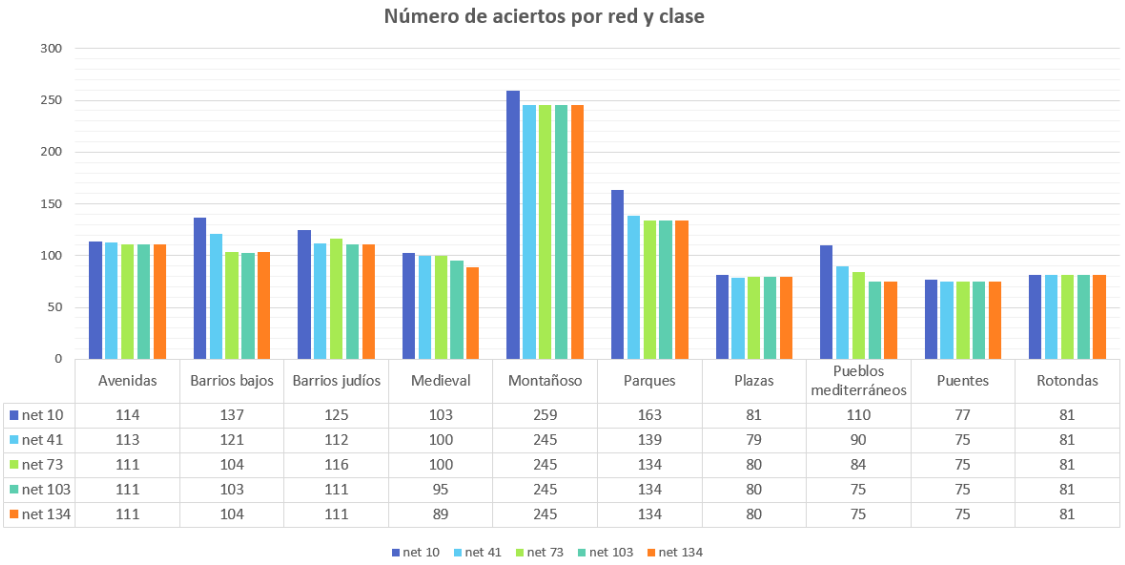
4.3.3. Resultados

Los mejores resultados, en número de imágenes, para todas las capas se muestran en la Figura 4.3. Los porcentajes y capas en los que se obtienen dichos aciertos se muestran en la Figura 4.4 y en la Tabla 4.5, usándose diferentes capas para cada par red-clase. Todos los porcentajes obtenidos para cada caso se encuentran en el Anexo A.3.

Red	Avenidas	Barrios bajos	Barrios judíos	Medieval	Montañoso	Parques	Plazas	Pueblos mediterráneos	Puentes	Rotondas
net 10	**57,29 %**	**49,82 %**	**39,43 %**	**55,38 %**	**83,82 %**	**46,57 %**	**46,02 %**	**48,67 %**	**64,71 %**	**66,94 %**
net 41	56,78 %	44,00 %	35,33 %	53,76 %	79,29 %	39,71 %	44,89 %	39,82 %	63,03 %	66,94 %
net 73	55,78 %	37,82 %	36,59 %	53,76 %	79,29 %	38,29 %	45,45 %	37,17 %	63,03 %	66,94 %
net 103	55,78 %	37,45 %	35,02 %	51,08 %	79,29 %	38,29 %	45,45 %	33,19 %	63,03 %	66,94 %
net 134	55,78 %	37,82 %	35,02 %	47,85 %	79,29 %	38,29 %	45,45 %	33,19 %	63,03 %	66,94 %

Tabla 4.5: Porcentajes de los casos de mayor aciertos por clase y red.

En la Figura 4.4 se codifican las capas y redes que devuelven un mejor resultado para cada clase, p.e. Para la clase Avenidas y la red net 10 se obtiene el mejor resultado extrayendo características de la capa $inception\_4a - 3x3 - reduce$, es decir, usando la combinación bajo el acrónimo 10 4a.



	Avenidas	Barrios bajos	Barrios judíos	Medieval	Montañoso	Parques	Plazas	Pueblos mediterráneos	Puentes	Rotondas
net 10	114	137	125	103	259	163	81	110	77	81
net 41	113	121	112	100	245	139	79	90	75	81
net 73	111	104	116	100	245	134	80	84	75	81
net 103	111	103	111	95	245	134	80	75	75	81
net 134	111	104	111	89	245	134	80	75	75	81

Figura 4.3: Histograma del número de aciertos por red y clase.

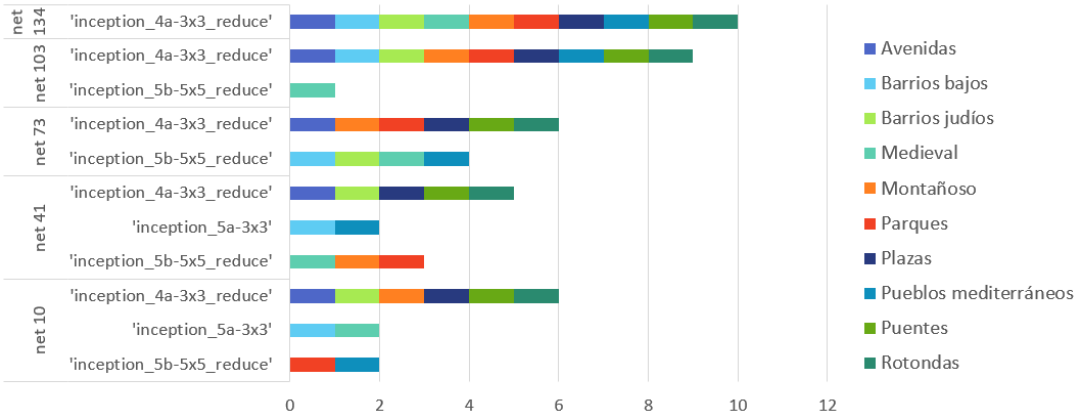


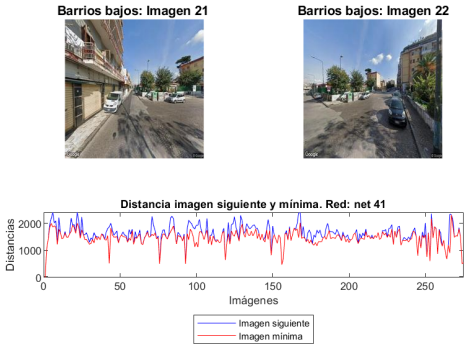
Figura 4.4: Histograma mejores capas por clase y red.

Resultados cualitativos Para ver más en detalle los resultados mostrados en la Figura 4.3, se muestran las figuras de la evolución de las distancias, en gráfica 2-D y a través de la matriz de distancias entre características, esta última se representa mediante la instrucción de Matlab *imagesc*.

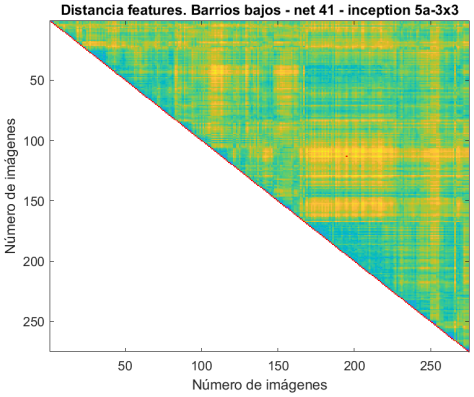
Nótese que por definición, los valores de la diagonal principal son siempre 0 (comparar las características de una imagen con ellas mismas). Para mejorar la calidad de los resultados, se altera la visualización estándar dada por *imagesc*, pintando de blanco los valores inferiores a la diagonal principal y la propia diagonal principal, de un color distinto al resto de colores que aparezcan en la imagen.

El escenario seleccionado para el ejemplo es la clase Barrios bajos. Se compararán dentro de una misma red, en este caso, *41 5a* y *41 out* (ver Tabla 4.4) para observar el comportamiento de las distancias en dos puntos diferentes de la misma red.

Como se puede ver en las Figuras 4.5 y 4.6, para capas intermedias de la red se obtienen mejores resultados en cuanto a la capacidad de organización de las imágenes.

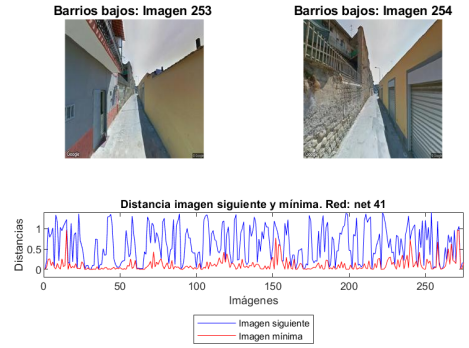


(a) Distancia imagen mínima y siguiente net 41, capa inception_5a-3x3. Porcentaje acierto = 44 %.

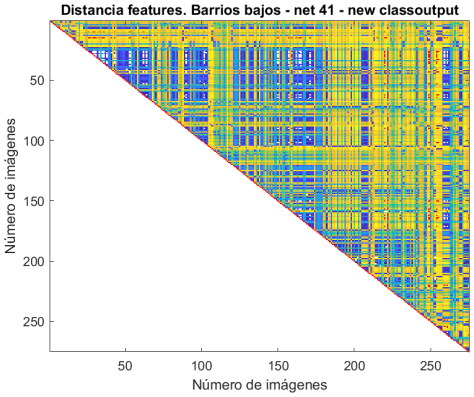


(b) Distancia entre características net 41, capa inception_5a-3x3.

Figura 4.5: Ejemplo distancias Barrios bajos net 41, capa inception_5a-3x3.



(a) Distancia imagen mínima y siguiente net 41, capa new_classoutput. Porcentaje acierto = 7,64, %.



(b) Distancia entre características net 41, capa new_classoutput.

Figura 4.6: Ejemplo distancias Barrios bajos net 41, capa new_classoutput.

En las Figuras 4.5a y 4.6a se representa la evolución de las distancias a la imagen siguiente (en azul) y la mínima distancia (en rojo). En estas imágenes también se representa la imagen anterior y la imagen siguiente a la máxima distancia encontrada en la curva de distancias a la imagen siguiente. Nótese la gran variación visual entre estas imágenes consecutivas.

En las Figuras 4.5b y 4.6b se ve la matriz de distancia entre características representada mediante el comando *imagesc* de Matlab, donde vemos la distribución de las distancias, en azul las más próximas y en amarillo las más lejanas.

4.3.4. Discusión

Tras el estudio de todos los casos de aciertos, se ha comprobado que la efectividad obtenida a través de las distancias entre características indica que, donde mayor probabilidad hay de que el valor de la distancia de la imagen siguiente coincida con la distancia

mínima de la imagen a estudio con el resto, se produce en capas intermedias de la red.

Creemos que esto es debido a que en las capas intermedias de la red, se obtienen resultados más enfocados hacia a las imágenes proporcionadas, y a medida que vamos avanzando en las capas de la red, se van obteniendo resultados más ajustados a la clase, y menos a la imagen.

Durante las pruebas realizadas, se ha comprobado que de entre las capas medias de la red, la capa con mayor éxito, para todas las redes, es la capa *inception_4a-3x3_reduce*. Esta capa se sitúa en la posición 43 de la red, lo que resulta en que en 3 de los 5 entrenamientos realizados, esta capa no ha sido entrenada.

Como se ve en la Tabla 4.5 el mejor resultado es en la clase Montañoso en la combinación con el acrónimo 10 4a capa-red (ver la Tabla 4.4).

4.4. Experimento 2: Agrupación

4.4.1. Objetivos

En este experimento se ha estudiado la viabilidad de establecer a través de las distancias entre características, agrupaciones espacio-temporales de los cuadros en cada trayectoria, de manera que los límites de estos clústers indiquen cambios de trayectoria del vehículo o persona que realiza la grabación. Para ello, relacionaremos las agrupaciones obtenidas con las anotadas en la sección 3.1.2.

4.4.2. Desarrollo y métricas de evaluación

Todos los cálculos realizados para establecer los *clústers* se han realizado usando los valores de características extraídos de la última capa de la red: *new classoutput*.

Una vez establecidos los casos de estudio, descritos en la sección 3.4, se procede a evaluar la bondad de los métodos de agrupación mediante el siguiente procedimiento:

1- En primer lugar se determina un valor de *clúster* máximo. Se establece el valor de 15 *clústers* como máximo.

2- Escogemos uno de los dos algoritmos utilizados (Kmeans o Linkage), una de las métricas de distancia propuestas (*sqEuclidean*, *cosine* o *correlation*) y una de las redes entrenadas (Ver Tabla 4.3).

3- Elegidas las opciones, se evalúa cada caso mediante la instrucción *evalcluster* de Matlab, la cual proporciona: número óptimo de *clústers*, el valor de la métrica de la agrupación, asignación de cada imagen a su *clúster* y valor de sus centroides.

El número óptimo de *clústers* viene determinado por la posición que ocupa el valor mínimo de la variable *CriterionValues* (métrica de agrupación) tras haber evaluado la trayectoria con cada criterio.

4- Al tener varios criterios diferentes, cada uno puede devolver un número óptimo de *clústers* distinto, por lo que para fijar el número de *clústers*, usamos dos criterios distintos y restamos los vectores *Criterion Values*. Ponemos en valor absoluto la diferencia

y se escoge la posición del valor mínimo del vector diferencia, es decir, el número de *clústers* que devuelva el valor más cercano entre dos criterios. Este número de *clústers* constituye una solución de compromiso entre los dos criterios. Este proceso se representa gráficamente en la Figura 4.7.

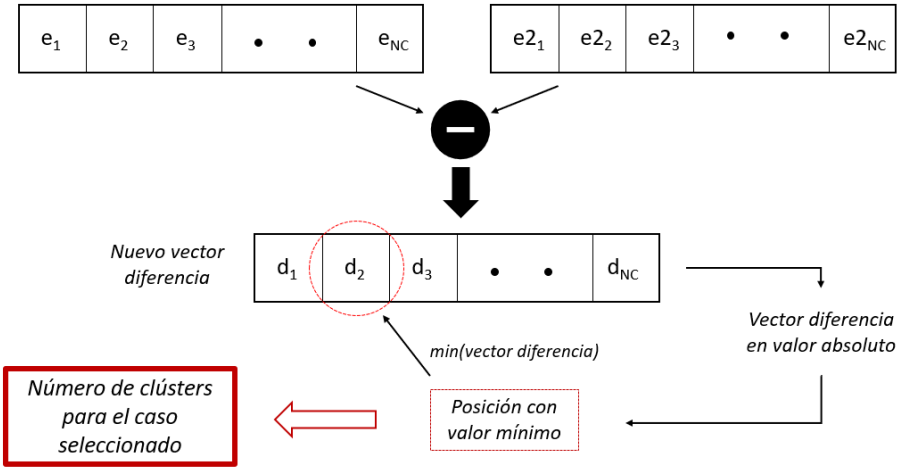


Figura 4.7: Pasos para la extracción del número de *clústers* utilizando la función *evalcluster*.

5- Una vez establecido el número de *clústers*, se asignan los datos a cada *clúster* mediante la función *clusterdata*.

6- La asignación de imágenes a los *clústers* no es continua, por consiguiente, se realiza un ajuste para que sí sea continua. Esto es, si por ejemplo las imágenes uno y dos pertenecen al *clúster* 3 y la imagen tres pertenece al *clúster* 7, con la ordenación de los *clústers* quedaría que las imágenes uno y dos forman parte del *clúster* 1 y la imagen 3 formaría parte del *clúster* 2.

7- Una vez realizado el ajuste para que los *clústers* sean continuos, representamos los *clústers* obtenidos sobre las trayectorias reales mediante el comando *scatter* de Matlab.

8- Para medir cuantitativamente la viabilidad de los resultados de los *clústers* se calcula la desviación típica y la divergencia respecto al número de *clústers* anotados.

4.4.3. Resultados

En total se evaluarán 360 posibilidades, escogiendo entre las 6 opciones de pares de criterios (ver sección 3.4), los dos algoritmos de *clustering* (Kmeans y Linkage), las 3 opciones de métricas de distancia (*sqEuclidean*, *cosine* y *correlation*) y las 10 clases.

Los resultados se mostrarán con estos acrónimos:

K - Kmeans **L** - Linkage **GT** - Ground Truth

DB - Davies-Bouldin **SH** - Silhouette

CH - Calinski-Harabasz **Gap** - Gap

sqE - SqEuclidean **cos** - Cosine **xcorr** - Correlation

En la Figura 4.8 se puede ver el resultado tras realizar el comando de Matlab *boxplot*. En esta representación podemos ver las desviaciones típicas (en azul) y la media (en rojo). Los mejores resultados son los que tengan una media lo más próxima al cero posible y menor desviación típica.

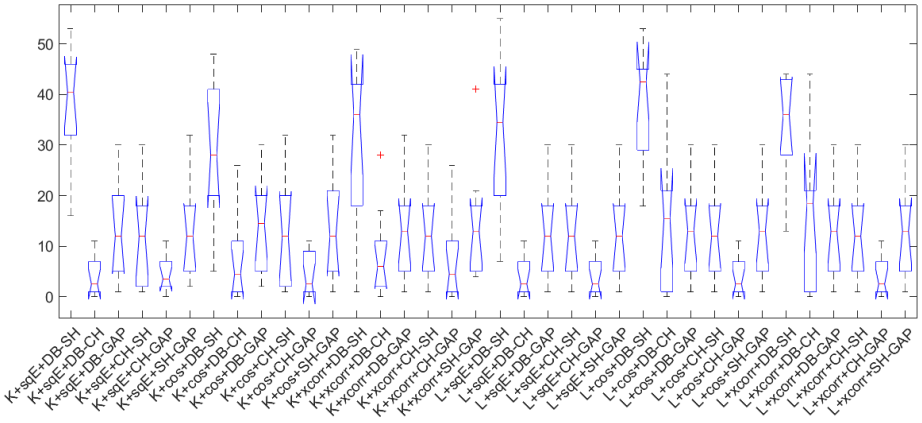


Figura 4.8: Representación en formato *boxplot* de la divergencia entre el número de agrupaciones obtenidas y anotadas.

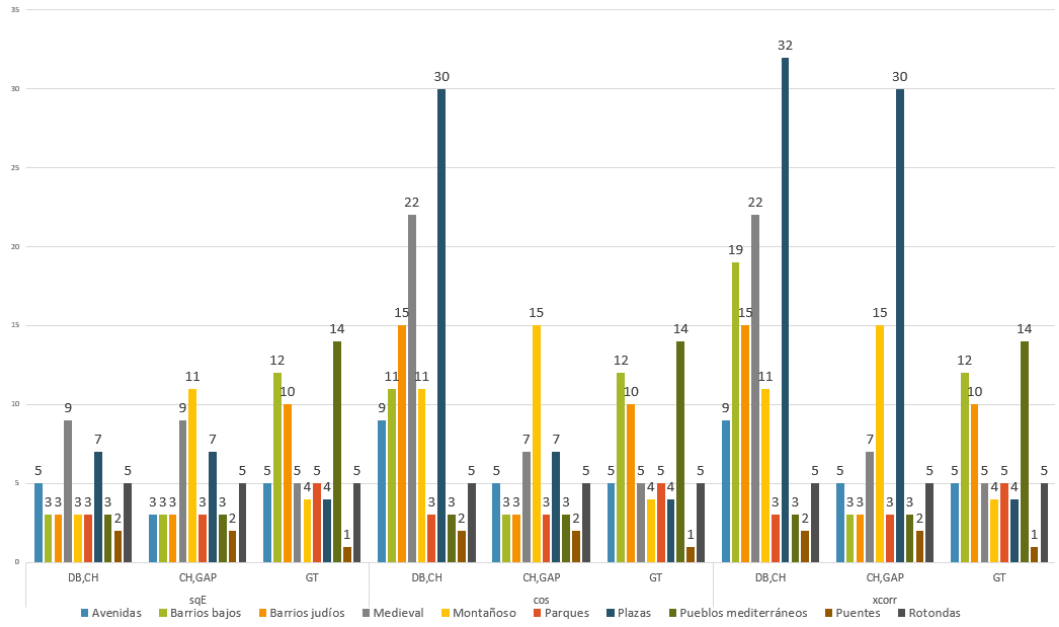


Figura 4.9: Histograma resultado del *clustering* para todos los casos, más el *ground truth*, con el algoritmo Kmeans.

A la vista de lo observado en la Figura 4.8, determinamos que las mejores opciones son las que se componen por los pares de criterios DB,CH y CH,GAP para el algoritmo Kmeans. Las siguientes representaciones de trayectorias se mostrarán sobre estos criterios.

En la Figura 4.9 se puede ver la representación de todos los valores obtenidos para los dos mejores casos mencionados, (DB,CH y CH,GAP) para el algoritmo Kmeans. En la gráfica también se muestra el *ground truth* (ver Figura 3.8), para visualizar cómo se parecen las distribuciones de los *clústers* obtenidos. En el Anexo B.1 se puede observar el histograma para los mismos valores usando el algoritmo de Linkage.

En las Figuras 4.10, 4.11 y 4.12 se pueden ver ejemplos de algunas de las trayectorias obtenidas, con las anotaciones de los *clústers* esperados en ellas en contraposición con los *clústers* obtenidos.

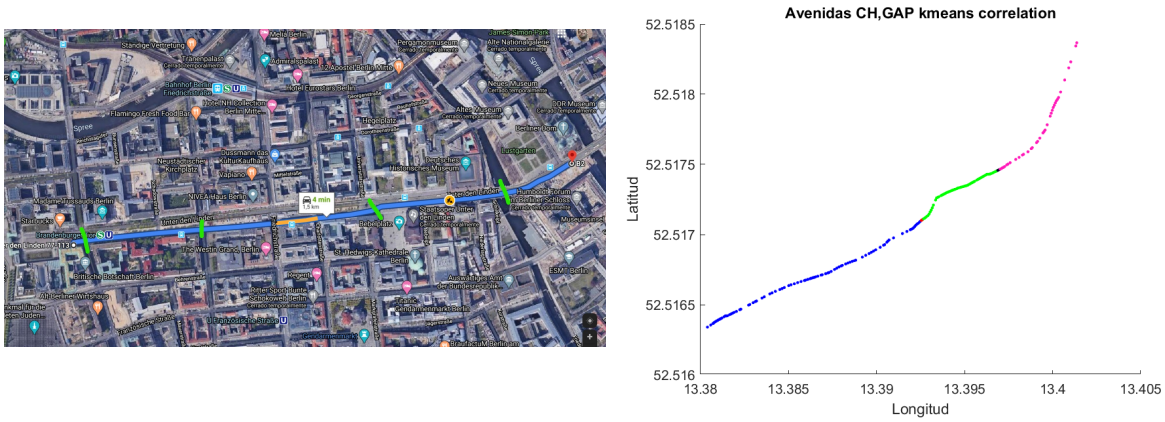


Figura 4.10: Trayectoria clase Avenidas extraída de Google Street View y representación de los *clústers* anotados: 5 y obtenidos: 5.

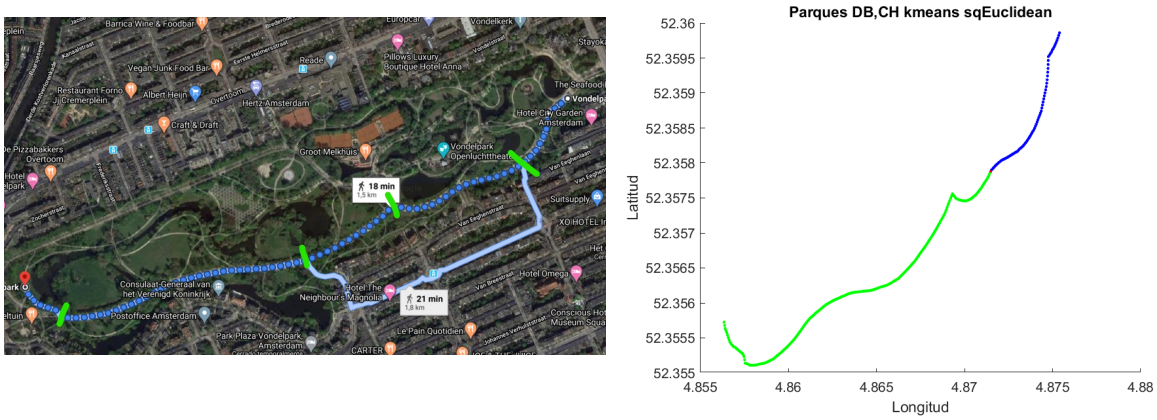


Figura 4.11: Trayectoria clase Parques extraída de Google Street View y representación de los *clústers* anotados: 5 y obtenidos: 3.

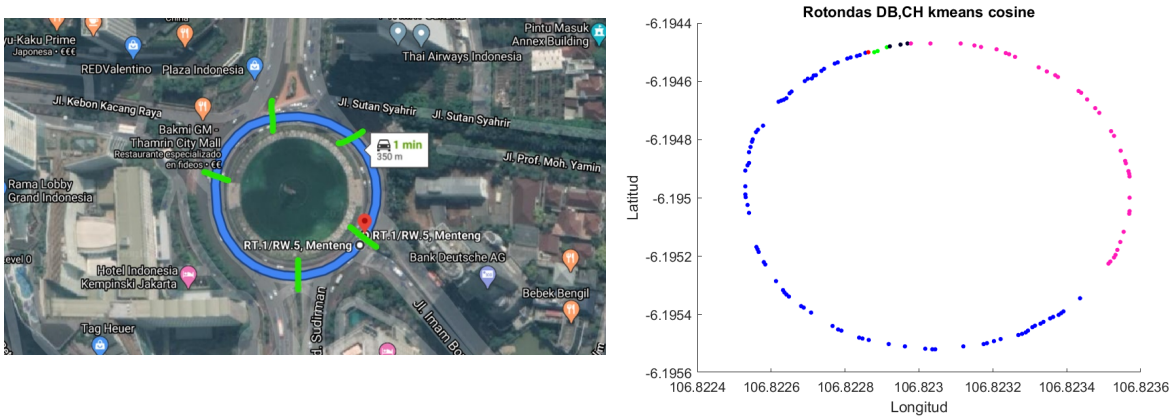


Figura 4.12: Trayectoria clase Rotondas extraída de Google Street View y representación de los *clústers* anotados: 5 y obtenidos: 5.

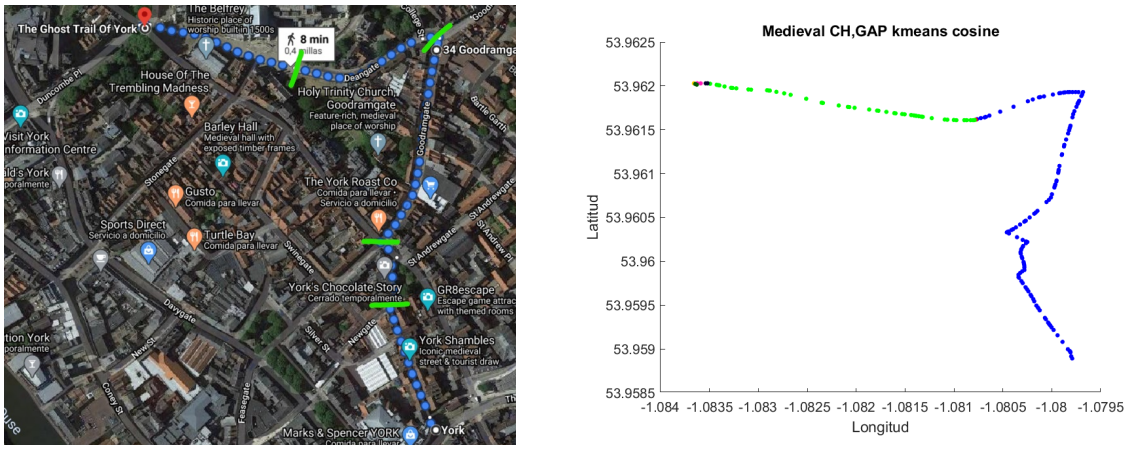


Figura 4.13: Trayectoria clase Medieval extraída de Google Street View y representación de los *clústers* anotados: 5 y obtenidos: 7.

En el Anexo B.2 se muestran algunos ejemplos más de la representación de las trayectorias con las anotaciones de los *clústers* anotados y los obtenidos.

4.4.4. Discusión

Según los datos observados, se determina que en numerosas trayectorias se obtienen unos resultados cercanos a los anotados en la primera hipótesis formulada (ver Figura 3.8).

El problema observado viene dado en el punto donde se produce el cambio de *clúster*. En el primer filtro diseñado, se indicó que las situaciones donde habría un cambio hipotético de *clúster* sería en giros y en cambios de manzana. Esto no es siempre así, puesto que no se produce el cambio de *clúster* bajo los escenarios descritos.

Lo que sí se ha observado es que en una gran cantidad de casos no se producen los cambios de *clúster* en los puntos donde se había previsto que pudieran suceder, pero sí en zonas próximas. Esto puede deberse a que al tratarse de un estudio a la salida de las redes neuronales, como se mencionó en las conclusiones del experimento anterior, los resultados se orientan más hacia la clasificación de la clase que de las propias imágenes. Esto puede derivar en que las percepciones espacio-temporales que somos capaces de detectar no lleguen a los niveles de abstracción a los que las redes sí pueden.

Entre todas las pruebas realizadas, la mejor de todas ellas es la que se realiza con el algoritmo Kmeans, métrica de distancia *sqEuclidean* y los criterios de Davies-Bouldin y Calinski-Harabasz (DB,CH).

Por tanto, se debe entender que en este experimento los resultados para algunos de los casos descritos son muy prometedores para obtener unos valores de *clustering* que se aproximen a lo anotado sobre las trayectorias reales, pero no son definitivos, dado que se han realizado sobre una muestra de datos pequeña dentro del escenario en el que está enmarcado este TFG.

5
Conclusiones

5.1. Conclusiones

Este TFG tenía como objetivos diseñar una base de datos rica en variedad de escenarios con la que, mediante el uso de CNNs y de las características que se pueden extraer de ellas, poder conseguir, mediante la realización de casos prospectivos, mostrar el potencial de los resultados.

Para ello, se llevó a cabo un estudio sobre las herramientas que serían de utilidad para la obtención de las imágenes que compondrían la base de datos, y de las cuales nos serviríamos para entrenar posteriormente las redes. Así mismo, se realizó un estudio sobre las tecnologías que permitirían sacar el máximo partido en términos de rendimiento en la ejecución de los entrenamientos y de extracción de características.

Se diseñó el conjunto de la base de datos para intentar que existiera bastante disparidad entre las distintas clases y trayectorias.

Se realizaron diversas pruebas con las que se hallaron los mejores hiperparámetros que se usarían para configurar, ajustar la arquitectura final de la red y conseguir los mejores resultados posibles en los entrenamientos.

Realizadas las pruebas y habiendo obtenido la red final, se efectuaron diversos entrenamientos para evaluar el funcionamiento de la red congelando determinados número de capas. Sobre estas redes, se seleccionaron distintas capas para la extracción de las características sobre las trayectorias descargadas.

A través de estas características se evaluó la capacidad de *clustering*, sobre las trayectorias reales descargadas, de las redes entrenadas y de las características extraídas obteniendo resultados prometedores.

Por consiguiente, podemos establecer que los objetivos principales marcados al inicio de este TFG, la creación de la base de datos y la obtención de *clústers* coherentes en concordancia con la primera hipótesis formuladas, se han llevado a cabo.

En cuanto a los resultados obtenidos a lo largo de la realización del TFG, se ha observado que aunque existe una concordancia en el número de *clústers* anotados y observados, la gran diferencia reside en el punto de la trayectoria donde aparecen los cambios de *clústers*. Creemos que una de las causas de estas discrepancias reside en cómo están muestreadas las trayectorias y en el posible ruido que, aún habiéndose realizado una limpieza de la base de datos, está presente en las imágenes.

5.2. Trabajo futuro

Como primer posible trabajo futuro a partir de este TFG, se propone establecer un estudio más extenso y exhaustivo, aumentando el número de clases y viendo como se ajustarían los nuevos resultados a ellas y teniendo otra arquitectura distinta de red, como puede ser AlexNet.

Otro añadido sería el desarrollo de un algoritmo que mejorase la precisión de la limpieza de la base de datos para impedir de una forma más fina la existencia de imágenes que rompan la homogeneidad de la trayectoria.

Otro posible trabajo sería dado un set de imágenes, a través de las características, ser capaces reordenar vídeos fehacientemente, detectando si en esa trayectoria faltan imágenes. Evaluar como se comportarían los *clústers* en cruces de dos trayectorias en repetidas ocasiones.

Partiendo de la idea que propone el *life logging*, se podrían grabar físicamente vídeos de rutas reales que se hagan a diario, para comprobar si habiendo pequeñas variaciones en los trayectos, como puede ser el uso de distinto medio de transporte, ir por aceras distintas de una misma calle, y estudiar si con estos pequeños cambios, al final el resultado de agrupaciones se comporta de una manera similar o si en su defecto los resultados serían dispares.

Bibliografía

[1] Cienciasfera, "Implicación de la informática en la traducción," https://www.um.es/docencia/barzana/TEI/Informatica-Aplicada-a-la-Traduccion-Introduccion.html, 2018, [Último acceso: 29 de junio de 2020].

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, 2016, pp. 2818–2826.

[3] R. Ng, "Optimization Algorithm: Mini-batch Stochastic Gradient Descent (SGD)," https://www.deeplearningwizard.com/deep_learning/boosting_models_pytorch/lr_scheduling/, 2019, [Último acceso: 30 de junio de 2020].

[4] deeplearning4j, "Early Stopping," https://jrmerwin.github.io/deeplearning4j-docs/earlystopping, 2017, [Último acceso: 30 de junio de 2020].

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[6] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, Columbus, Ohio, 2014, pp. 806–813.

[7] C. Piech, "K Means,The Basic Idea," https://stanford.edu/~cpiech/cs221/handouts/kmeans.html, 2013, [Último acceso: 29 de junio de 2020].

[8] A. Ihler, "Clustering (2): Hierarchical Agglomerative Clustering," https://www.youtube.com/watch?v=0coE7JlbXvY, 2015, [Último acceso: 15 de junio de 2020].

[9] Google Developers, "Fuentes de las fotografías," https://www.google.com/streetview/explore/, 2020, [Último acceso: 29 de junio de 2020].

[10] A. H. Andrew, K. Eustice, and A. Hickl, "Using location lifelogs to make meaning of food and physical activity behaviors," in *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*. IEEE, 2013, pp. 408–411.

[11] D. J. Matich, "Redes neuronales: Conceptos básicos y aplicaciones," *Universidad Tecnológica Nacional, México*, 2001.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, Santiago, Chile, 2015, pp. 1026–1034.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[14] M. Boden, "A guide to recurrent neural networks and backpropagation," *the Dallas project*, 2002.

[15] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[16] J. Brownlee, *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018.

[17] N. Nain, S. Vipparthi, and B. Raman, *Computer Vision and Image Processing: 4th International Conference, CVIP 2019, Jaipur, India, September 27–29, 2019, Revised Selected Papers, Part I*, ser. Communications in Computer and Information Science. Springer Singapore, 2020. [Online]. Available: https://books.google.es/books?id=c5DZDwAAQBAJ

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Published as a conference paper at ICLR 2015*, 2014.

[19] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *Advances in Neural Information Processing Systems*, California, 2017, pp. 1731–1741.

[20] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation." in *AAAI*, 2020, pp. 13 001–13 008.

[21] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[22] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[23] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 12, pp. 1650–1654, 2002.

[24] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

[25] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[26] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[27] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[28] Google Developers, "Street view publish api," `https://developers.google.com/streetview/publish/reference/rest`, 2019, [Último acceso: 29 de junio de 2020].

[29] Google Developers, "Directions api," `https://developers.google.com/maps/documentation/directions/intro`, 2019, [Último acceso: 29 de junio de 2020].

A

Tablas

A.1. Trayectorias extraídas

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Avenidas	9 de Julio, Buenos Aires	-34.609912,-58.381114	-34.591857,-58.381979	2	0,18	DRIVING
Avenidas	Campos Eliseos, Paris	48.869336,2.309251	48.8735548,2.2960899	1,1	0,32	DRIVING
Avenidas	Broadway, Nueva York	40.811944,-73.960873	40.827352,-73.949774	2	0,18	DRIVING
Avenidas	Unter den Linden, Berlin	52.516272,13.380397	52.518368,13.401444	1,5	0,23	DRIVING
Avenidas	Las Vegas Blvd S, Las Vegas	36.107305,-115.172796	36.125010,-115.169520	2,1	0,17	DRIVING
Avenidas	Andrassy UT, Budapest	47.500024,19.055602	47.513532,19.075817	2,1	0,17	DRIVING
Avenidas	Gran Vía, Madrid	40.424707,-3.711998	40.419113,-3.697486	1,5	0,23	DRIVING
Avenidas	Avda Paulista, São Paulo	-23.557927,-46.660605	-23.571457,-46.644144	2,2	0,16	DRIVING
Avenidas	5th Ave, Nueva York	40.763287,-73.973740	40.743545,-73.988143	2,6	0,13	DRIVING
Avenidas	Tverskaya St, Moscú	55.757120,37.614837	55.771152,37.594192	2	0,18	DRIVING
Avenidas	Oxford St, Londres	51.513511,-0.157869	51.516441,-0.130634	2,2	0,16	DRIVING

Tabla A.1: Avenidas

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Barrios Bajos	Secondigliano, Italia	40.895004, 14.264719	40.889122, 14.274641	1,4	0,25	WALKING
Barrios Bajos	Secondigliano2, Italia	40.889122, 14.274641	40.884718, 14.273036	0,6	0,58	WALKING
Barrios Bajos	Favela de Rocinha, Brasil	-22.989620, -43.252309	-22.986763, -43.245522	1,6	0,22	DRIVING
Barrios Bajos	Molenbeek, Bélgica	50.865881, 4.340658	50.853594, 4.328812	2,1	0,17	DRIVING
Barrios Bajos	Roubaix, Francia	50.700520, 3.183000	50.688568, 3.172182	2	0,18	WALKING
Barrios Bajos	Martim Moniz e Intendente, Lisboa	38.726295, -9.134563	38.720908, -9.131809	0,7	0,50	WALKING
Barrios Bajos	Martim Moniz e Intendente2, Lisboa	38.720908, -9.131809	38.713235, -9.130105	1,1	0,32	WALKING
Barrios Bajos	Rosengård, Suecia	55.589880, 13.038208	55.577616, 13.043171	2	0,18	WALKING
Barrios Bajos	Ferentari, Rumanía	44.401682, 26.077554	44.389819, 26.077006	1,8	0,19	DRIVING
Barrios Bajos	Las 3000 viviendas, España	37.357409, -5.971586	37.364988, -5.965844	1,8	0,19	DRIVING
Barrios Bajos	La Courneuve, Francia	48.928390, 2.390761	48.932156, 2.389380	0,65	0,54	WALKING
Barrios Bajos	La Courneuve2, Francia	48.932156, 2.389380	48.932622, 2.399834	0,9	0,39	WALKING
Barrios Bajos	Ciudad Meridiana, Barcelona	41.463220, 2.168541	41.460676, 2.175626	1	0,35	WALKING
Barrios Bajos	Ciudad Meridiana2, Barcelona	41.460676, 2.175626	41.463499, 2.178659	0,45	0,78	WALKING
Barrios Bajos	Khlong Toei, Bangkok	13.717474, 100.559717	13.723446, 100.563483	1,1	0,32	DRIVING

Tabla A.2: Barrios bajos

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Barrios judíos	Kazimierz, Cracovia	50.048321, 19.941583	50.050825, 19.944935	1,3	0,27	DRIVING
Barrios judíos	Williamsburg, Nueva York	40.705916, -73.963023	40.704078, -73.962896	0,8	0,44	DRIVING
Barrios judíos	Williamsburg2, Nueva York	40.704078, -73.962896	40.702715, -73.954583	1,28	0,27	DRIVING
Barrios judíos	Erzsébetváros, Budapest	47.503949, 19.069968	47.498829, 19.059924	1,8	0,19	DRIVING
Barrios judíos	Erzsébetváros2, Budapest	47.498829, 19.059924	47.496120, 19.060050	0,7	0,50	DRIVING
Barrios judíos	Hervás, Cáceres	40.275201, -5.855646	40.275864, -5.857089	0,3	1,17	WALKING
Barrios judíos	Balat, Estambul	41.026525, 28.948962	41.031996, 28.946441	0,85	0,41	WALKING
Barrios judíos	Josefov, Praga	50.083618, 14.420177	50.090321, 14.420690	0,9	0,39	WALKING
Barrios judíos	Pletzl, París	48.856056, 2.364034	48.862762, 2.357783	1,3	0,27	DRIVING
Barrios judíos	Berlín, Alemania	52.527488, 13.408493	52.523661, 13.399095	1,2	0,29	DRIVING
Barrios judíos	Berlín2, Alemania	52.523661, 13.399095	52.523514, 13.390714	0,7	0,50	DRIVING
Barrios judíos	Córdoba	37.881906, -4.774740	37.880711, -4.783945	1,1	0,32	WALKING
Barrios judíos	Mea Shearim, Jerusalén	31.786186, 35.216386	31.786394, 35.221479	1,4	0,25	DRIVING
Barrios judíos	Mea Shearim2, Jerusalén	31.786394, 35.221479	31.790236, 35.223992	0,7	0,50	DRIVING
Barrios judíos	Lucena, España	37.408838, -4.479522	37.409582, -4.486898	0,7	0,50	WALKING

Tabla A.3: Barrios judíos

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Medieval	Olite, España	42.481387, -1.652146	42.480314, -1.648630	0,4	0,88	WALKING
Medieval	Carcasona, Francia	43.206557, 2.365995	43.208006, 2.363963	0,35	1,00	WALKING
Medieval	Carcasona2, Francia	43.208006, 2.363963	43.205477, 2.362889	0,35	1,00	WALKING
Medieval	San Gimignano, Italia	43.466246, 11.042761	43.470571, 11.041826	0,6	0,58	WALKING
Medieval	Besalú, Girona	42.197803, 2.697804	42.199873, 2.700431	0,55	0,64	DRIVING
Medieval	Dinant, Bélgica	50.260220, 4.913501	50.256742, 4.913794	0,5	0,70	WALKING
Medieval	Dinant2, Bélgica	50.256742, 4.913794	50.254291, 4.917262	0,45	0,78	WALKING
Medieval	Edimburgo	55.952800, -3.193500	55.953433, -3.200634	0,64	0,55	DRIVING
Medieval	Edimburgo2	55.953433, -3.200634	55.953450, -3.206897	0,48	0,73	DRIVING
Medieval	Eze, Francia	43.728084, 7.361451	43.727471, 7.361715	0,18	1,94	WALKING
Medieval	Eze2, Francia	43.727471, 7.361715	43.728122, 7.362181	0,1	3,50	WALKING
Medieval	Mont Saint Michel, Normandia	48.634952, -1.511308	48.635659, -1.510265	0,15	2,33	WALKING
Medieval	Mont Saint Michel2, Normandia	48.635659, -1.510265	48.636660, -1.510756	0,15	2,33	WALKING
Medieval	Vanalinn, Tallin	59.442702, 24.749410	59.439954, 24.746260	0,4	0,88	WALKING
Medieval	Vanalinn2, Tallin	59.439954, 24.746260	59.435943, 24.738757	0,7	0,50	WALKING
Medieval	York, Inglaterra	53.958865, -1.079732	53.961833, -1.079717	0,32	1,09	WALKING
Medieval	York2, Inglaterra	53.961833, -1.079717	53.962000, -1.083591	0,32	1,09	WALKING
Medieval	Monsanto, Portugal	40.038833, -7.116858	40.039063, -7.114725	0,35	1,00	DRIVING
Medieval	Monsanto2, Portugal	40.039063, -7.114725	40.038824, -7.116957	0,35	1,00	DRIVING

Tabla A.4: Medieval

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Montañoso	Zermatt, Suiza	46.018292, 7.749247	46.027377, 7.753201	1,2	0,29	WALKING
Montañoso	Chambéry, Francia	45.565331, 5.925824	45.569774, 5.912371	1,5	0,23	DRIVING
Montañoso	Metsovo, Grecia	39.772728, 21.180589	39.769813, 21.185551	1,3	0,27	DRIVING
Montañoso	Potes, España	43.154724, -4.626293	43.154724, -4.620294	0,65	0,54	WALKING
Montañoso	Potes2, España	43.154724, -4.620294	43.152408, -4.622925	0,35	1,00	WALKING
Montañoso	San Romá de les Bons, Andorra	42.534448, 1.576910	42.534188, 1.588087	1,2	0,29	DRIVING
Montañoso	Viella, Lérida	42.704821, 0.789933	42.702020, 0.800973	1,3	0,27	WALKING
Montañoso	Lauterbrunnen, Suiza	46.599037, 7.907484	46.590228, 7.911944	1,3	0,27	DRIVING
Montañoso	Canazei, Italia	46.478654, 11.766987	46.475148, 11.774895	1,4	0,25	DRIVING
Montañoso	Conques, Francia	44.598911, 2.402945	44.598590, 2.394414	0,7	0,50	DRIVING
Montañoso	Scanno, Italia	41.903376, 13.881086	41.904395, 13.879568	1,2	0,29	DRIVING
Montañoso	Candelario, Salamanca	40.365010, -5.739558	40.368492, -5.743526	0,6	0,58	DRIVING

Tabla A.5: Montañoso

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Parques	Central Park, Nueva York	40.769657, -73.971948	40.783974, -73.967332	2	0,18	WALKING
Parques	Hyde Park, Londres	51.510555, -0.163738	51.503453, -0.177520	2	0,18	WALKING
Parques	Jardines de Luxemburgo, París	48.847528, 2.339634	48.845865, 2.334837	0,55	0,64	WALKING
Parques	Jardines de Luxemburgo2, París	48.845865, 2.334837	48.848640, 2.333255	0,45	0,78	WALKING
Parques	Ueno Park, Tokio	35.711080, 139.773683	35.717115, 139.775395	0,75	0,47	WALKING
Parques	Stanley Park, Vancouver	49.303000, -123.140050	49.307121, -123.153610	1,5	0,23	WALKING
Parques	Vondelpark, Ámsterdam	52.359832, 4.875439	52.355773, 4.856300	1,5	0,23	WALKING
Parques	Parque del Retiro, Madrid	40.420525, -3.680175	40.410161, -3.687988	1,7	0,21	WALKING
Parques	Parque Metropolitano, Santiago de Chile	-33.425025, -70.631924	-33.415415, -70.621272	2,4	0,15	DRIVING
Parques	Parque Yoyogi, Tokio	35.669069, 139.700363	35.674055, 139.695414	1,1	0,32	WALKING
Parques	Parque da Cidade, Oporto	41.171097, -8.680275	41.165656, -8.672752	1,2	0,29	WALKING

Tabla A.6: Parques

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Plazas	Plaza de San Marcos, Venecia	45.434718, 12.339116	45.433537, 12.337429	0,22	1,59	WALKING
Plazas	Plaza de San Marcos2, Venecia	45.433523, 12.337567	45.434672, 12.339157	0,23	1,52	WALKING
Plazas	Staromestké Namesti, Praga	50.088128, 14.421339	50.087060, 14.421214	0,15	2,33	WALKING
Plazas	Staromestké Namesti2, Praga	50.088082, 14.421223	50.087064, 14.421017	0,15	2,33	WALKING
Plazas	Grand-Place, Bruselas	50.846534, 4.353193	50.846860, 4.351910	0,13	2,69	WALKING
Plazas	Grand-Place2, Bruselas	50.846620, 4.353143	50.846940, 4.351928	0,11	3,18	WALKING
Plazas	Plaza Roja, Moscú	55.753272, 37.621686	55.755001, 37.619269	0,29	1,21	WALKING
Plazas	Plaza Roja2, Moscú	55.755097, 37.619028	55.753284, 37.621572	0,3	1,17	WALKING
Plazas	Plaza Mayor, Salamanca	40.964644, -5.663874	40.965314, -5.664370	0,11	3,18	WALKING
Plazas	Plaza Mayor2, Salamanca	40.964732, -5.663833	40.965401, -5.664297	0,12	2,92	WALKING
Plazas	Plaza de Mayo, Buenos Aires	-34.607820, -58.371303	-34.608839, -58.371015	0,5	0,70	DRIVING
Plazas	Trafalgar Square, Londres	51.508285, -0.128949	51.507822, -0.127240	0,2	1,75	WALKING
Plazas	Trafalgar Square2, Londres	51.507920, -0.127243	51.508361, -0.128944	0,2	1,75	WALKING
Plazas	Rynek Główny, Cracovia	50.060553, 19.937699	50.062593, 19.937186	0,3	1,17	WALKING
Plazas	Rynek Główny2, Cracovia	50.062643, 19.936859	50.060670, 19.937475	0,3	1,17	WALKING
Plazas	Plaza de la Constitución, Ciudad de México	19.433333, -99.132743	19.433311, -99.132630	0,65	0,54	DRIVING
Plazas	Plaza Mayor, Madrid	40.415658, -3.707525	40.415160, -3.707249	0,15	2,33	WALKING
Plazas	Plaza Mayor2, Madrid	40.415163, -3.707160	40.415669, -3.707410	0,14	2,50	WALKING
Plazas	Plaza de la Corredera, Córdoba	37.883341, -4.774199	37.883830, -4.774759	0,15	2,33	WALKING
Plazas	Plaza de la Corredera2, Córdoba	37.883833, -4.774865	37.883343, -4.774278	0,15	2,33	WALKING
Plazas	Piazza Navona, Roma	41.898627, 12.472796	41.899022, 12.473308	0,26	1,35	WALKING
Plazas	Piazza Navona2, Roma	41.899144, 12.473296	41.898785, 12.472880	0,27	1,30	WALKING

Tabla A.7: Plazas

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Pueblos mediterráneos	Rethymnon, Creta	35.371630, 24.473940	35.364609, 24.483010	2,9	0,12	DRIVING
Pueblos mediterráneos	Menton, Francia	43.774237, 7.494341	43.777184, 7.506650	2,5	0,14	DRIVING
Pueblos mediterráneos	Cefalú, Sicilia	38.040520, 14.025038	38.039582, 14.023557	1,3	0,27	DRIVING
Pueblos mediterráneos	Fornells, Menorca	40.049617, 4.126094	40.057301, 4.132414	1,6	0,22	DRIVING
Pueblos mediterráneos	Piran, Eslovenia	45.526117, 13.570542	45.530166, 13.563037	0,9	0,39	WALKING
Pueblos mediterráneos	La Valeta, Malta	35.898645, 14.507853	35.899048, 14.517775	1,4	0,25	DRIVING
Pueblos mediterráneos	Conil, Cadiz	36.277750, -6.090085	36.281424, -6.086942	1,3	0,27	DRIVING
Pueblos mediterráneos	Galipolli, Italia	40.053768, 17.973421	40.054338, 17.975979	0,55	0,64	DRIVING
Pueblos mediterráneos	Galipolli2, Italia	40.054338, 17.975979	40.055296, 17.976032	0,5	0,70	DRIVING
Pueblos mediterráneos	La Canea, Grecia	35.516579, 24.022843	35.516335, 24.016531	1	0,35	DRIVING
Pueblos mediterráneos	Antibes, Francia	43.579816, 7.119286	43.582457, 7.128340	1	0,35	WALKING
Pueblos mediterráneos	Motril, España	36.753342, -3.512255	36.749986, -3.515721	0,55	0,64	WALKING
Pueblos mediterráneos	Motril2, España	36.749986, -3.515721	36.747554, -3.516528	0,35	1,00	WALKING

Tabla A.8: Pueblos mediterráneos

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Puentes	Viaducto de Millau, Francia	44.073510, 3.023658	44.090950, 3.021266	1,9	0,18	DRIVING
Puentes	Puentes del Bósforo, Turquia	50.086078, 14.414265	50.086865, 14.408770	0,4	0,88	DRIVING
Puentes	Charles Bridge, Praga	50.086104, 14.414133	50.086862, 14.408795	0,4	0,88	WALKING
Puentes	Sydney Harbour Bridge	-33.855570, 151.208574	-33.848762, 151.212561	0,85	0,41	DRIVING
Puentes	Puente de las Cadenas, Budapest	47.498457, 19.041029	47.499408, 19.045929	0,4	0,88	DRIVING
Puentes	Puente de Brooklyn, Nueva York	40.702270, -73.991928	40.709079, -74.000508	1,15	0,30	DRIVING
Puentes	Golden Gate, San Francisco	37.809306, -122.477291	37.827214, -122.479370	1,9	0,18	DRIVING
Puentes	Tower Bridge, Londres	51.5041219,-0.0763404	51.506879, -0.074417	0,4	0,88	DRIVING
Puentes	Pont du Gard, Francia	43.946820, 4.536183	43.947724, 4.534708	0,2	1,75	WALKING
Puentes	Puente de Dom Luis, Oporto,	41.139072, -8.609193	41.140748, -8.609714	0,2	1,75	DRIVING
Puentes	Puente 25 de Abril, Lisboa	38.679883, -9.174873	38.697894, -9.178819	2	0,18	DRIVING

Tabla A.9: Puentes

Clase	Lugar	Origen	Destino	Distancia (km)	fpx	Mode
Rotondas	Columbus Circle, Nueva York	40.768060, -73.981384	40.768010, -73.981406	0,32	1,09	DRIVING
Rotondas	Plaza de la Cibeles, Madrid	40.419129, -3.693486	40.419183, -3.693575	0,2	1,75	DRIVING
Rotondas	Plaza de Castilla, Madrid	40.465618, -3.689532	40.465638, -3.689600	0,3	1,17	DRIVING
Rotondas	Grand Rond, Toulouse	43.595482, 1.454024	43.595345, 1.453956	0,7	0,50	DRIVING
Rotondas	Plaça dEspanya, Barcelona	41.375379, 2.148801	41.375393, 2.148860	0,3	1,17	DRIVING
Rotondas	The Magic Roundabout, Swindon	51.562570, -1.771547	51.562641, -1.771231	0,16	2,19	DRIVING
Rotondas	Arco del Triunfo, París	48.873819, 2.295830	48.873606, 2.295767	0,35	1,00	DRIVING
Rotondas	Fountain of Wealth Roundabout, Suntec City	1.294545, 103.859260	1.294614, 103.859281	0,26	1,35	DRIVING
Rotondas	Putrajaya, Malasia	2.934704, 101.691771	2.934823, 101.691883	0,55	0,64	DRIVING
Rotondas	The Hotel Indonesia Roundabout in Jakarta, Indonesia	-6.195319, 106.823463	-6.195255, 106.823502	0,35	1,00	DRIVING

Tabla A.10: Rotondas

A.2. Resultados agrupamiento

A.2.1. Kmeans

Métrica de distancia	DB,SH	DB,CH	DB,GAP	CH,SH	CH,GAP	SH,GAP
Avenidas	51	5	7	7	3	7
Barrios bajos	65	3	11	11	3	19
Barrios judios	59	3	15	15	3	15
Medieval	25	9	21	21	9	21
Montañoso	50	3	15	15	11	15
Parques	37	3	35	35	3	37
Plazas	48	7	30	22	7	22
Pueblos mediterraneos	51	3	9	15	3	9
Puentes	17	2	14	14	2	14
Rotondas	40	5	25	25	5	25

Tabla A.11: Resultado kmeans *clustering sqEuclidean*

Métrica de distancia	DB,SH	DB,CH	DB,GAP	CH,SH	CH,GAP	SH,GAP
Avenidas	34	9	7	7	5	7
Barrios bajos	60	11	19	11	3	11
Barrios judios	31	15	15	15	3	15
Medieval	25	22	21	21	7	21
Montañoso	31	11	31	15	15	15
Parques	46	3	35	37	3	37
Plazas	48	30	22	32	7	30
Pueblos mediterraneos	9	3	9	15	3	9
Puentes	19	2	14	14	2	14
Rotondas	40	5	25	25	5	26

Tabla A.12: Resultado kmeans *clustering cosine*

Métrica de distancia	DB,SH	DB,CH	DB,GAP	CH,SH	CH,GAP	SH,GAP
Avenidas	49	9	7	7	5	9
Barrios bajos	11	19	11	11	3	19
Barrios judios	59	15	15	15	3	15
Medieval	23	22	21	21	7	22
Montañoso	33	11	15	15	15	15
Parques	47	3	37	35	3	46
Plazas	44	32	22	22	30	22
Pueblos mediterraneos	51	3	9	9	3	9
Puentes	19	2	16	14	2	16
Rotondas	40	5	25	25	5	26

Tabla A.13: Resultado kmeans *clustering correlation*

A.2.2. Linkage

Métrica de distancia	DB,SH	DB,CH	DB,GAP	CH,SH	CH,GAP	SH,GAP
Avenidas	34	5	7	7	3	7
Barrios bajos	19	3	11	11	3	11
Barrios judios	56	3	15	15	3	15
Medieval	25	9	21	21	7	21
Montañoso	44	3	15	15	11	15
Parques	46	3	35	35	5	35
Plazas	59	7	22	22	7	22
Pueblos mediterraneos	56	3	9	9	3	9
Puentes	19	2	14	14	2	14
Rotondas	27	5	25	25	5	25

Tabla A.14: Resultado linkage *clustering sqEuclidean*

Métrica de distancia	DB,SH	DB,CH	DB,GAP	CH,SH	CH,GAP	SH,GAP
Avenidas	49	49	9	7	3	9
Barrios bajos	55	31	19	11	3	19
Barrios judios	56	23	15	15	3	15
Medieval	26	23	21	21	7	21
Montañoso	33	11	15	15	11	15
Parques	50	5	35	35	5	35
Plazas	57	30	22	22	7	22
Pueblos mediterraneos	56	15	13	9	3	13
Puentes	19	2	16	14	2	16
Rotondas	35	26	26	25	5	26

Tabla A.15: Resultado linkage *clustering cosine*

Métrica de distancia	DB,SH	DB,CH	DB,GAP	CH,SH	CH,GAP	SH,GAP
Avenidas	49	49	9	7	3	9
Barrios bajos	55	31	19	11	3	19
Barrios judios	54	31	15	15	3	15
Medieval	26	23	21	21	7	21
Montañoso	33	3	15	15	11	15
Parques	47	5	35	35	5	35
Plazas	32	30	22	22	7	22
Pueblos mediterraneos	56	15	13	9	3	13
Puentes	14	2	16	14	2	16
Rotondas	35	26	26	25	5	26

Tabla A.16: Resultado linkage *clustering correlation*

A.3. Resultados completos aciertos

	Numero Imágenes	Avenidas	Barrios bajos	Barrios judios	Medieval	Montañoso	Parques	Plazas	Pueblos mediterraneos	Puentes	Rotondas
		199	*275*	*317*	*186*	*309*	*350*	*176*	*226*	*119*	*121*
net 10	new_classoutput	12,56 %	7,27 %	5,68 %	12,37 %	9,06 %	6,29 %	14,20 %	4,42 %	14,29 %	14,05 %
	new_prob	12,56 %	7,27 %	5,68 %	12,37 %	9,06 %	6,29 %	14,20 %	4,42 %	14,29 %	14,05 %
	new_fully	29,65 %	18,91 %	16,40 %	32,26 %	36,57 %	12,57 %	28,41 %	14,60 %	29,41 %	48,76 %
	new_dropout	48,74 %	36,36 %	28,71 %	46,77 %	73,14 %	31,43 %	35,80 %	35,40 %	38,66 %	62,81 %
	inception_5b-5x5_reduce	53,27 %	47,27 %	36,28 %	53,23 %	83,17 %	46,57 %	39,77 %	48,67 %	54,62 %	66,12 %
	inception_5a-3x3	54,27 %	49,82 %	33,12 %	55,38 %	80,26 %	46,29 %	39,77 %	46,02 %	53,78 %	64,46 %
	inception_4c-5x5	51,26 %	38,91 %	30,28 %	48,92 %	72,17 %	38,86 %	38,07 %	33,19 %	57,14 %	62,81 %
	inception_4a-3x3_reduce	57,29 %	42,91 %	39,43 %	53,76 %	83,82 %	46,00 %	46,02 %	38,94 %	64,71 %	66,94 %
net 41	new_classoutput	12,06 %	7,64 %	6,31 %	13,98 %	7,12 %	4,57 %	13,07 %	4,87 %	13,45 %	9,92 %
	new_prob	12,06 %	7,64 %	6,31 %	13,98 %	7,12 %	4,57 %	13,07 %	4,87 %	13,45 %	9,92 %
	new_fully	31,16 %	17,82 %	15,46 %	33,33 %	33,33 %	14,85 %	26,14 %	11,50 %	26,05 %	42,98 %
	new_dropout	51,76 %	30,91 %	27,44 %	45,70 %	64,72 %	26,57 %	36,36 %	33,19 %	54,62 %	63,64 %
	inception_5b-5x5_reduce	53,27 %	40,73 %	35,33 %	53,76 %	79,29 %	39,71 %	40,34 %	38,50 %	56,30 %	63,64 %
	inception_5a-3x3	53,77 %	44,00 %	33,12 %	52,15 %	76,05 %	37,71 %	39,77 %	39,82 %	57,14 %	63,64 %
	inception_4c-5x5	48,74 %	30,18 %	25,24 %	47,31 %	67,96 %	36,29 %	38,61 %	27,88 %	52,94 %	58,68 %
	inception_4a-3x3_reduce	56,78 %	39,64 %	35,33 %	48,92 %	79,29 %	39,14 %	44,89 %	33,19 %	63,03 %	66,94 %
net 73	new_classoutput	10,05 %	6,18 %	5,05 %	9,68 %	10,68 %	4,29 %	10,80 %	7,08 %	8,40 %	8,26 %
	new_prob	10,05 %	6,18 %	5,05 %	9,68 %	10,68 %	4,29 %	10,80 %	7,08 %	8,04 %	8,26 %
	new_fully	25,63 %	16,36 %	14,83 %	32,80 %	30,10 %	12,86 %	23,30 %	12,83 %	27,73 %	40,50 %
	new_dropout	45,73 %	31,27 %	26,18 %	47,31 %	63,11 %	23,71 %	39,20 %	24,78 %	55,46 %	61,98 %
	inception_5b-5x5_reduce	51,76 %	37,82 %	36,59 %	53,76 %	77,67 %	36,29 %	38,64 %	37,17 %	54,62 %	66,12 %
	inception_5a-3x3	53,27 %	36,36 %	36,59 %	52,15 %	75,08 %	36,29 %	38,64 %	36,28 %	52,10 %	64,46 %
	inception_4c-5x5	50,25 %	25,45 %	23,03 %	46,77 %	63,43 %	27,71 %	35,80 %	22,57 %	49,58 %	61,98 %
	inception_4a-3x3_reduce	55,78 %	37,45 %	35,02 %	47,85 %	79,29 %	38,29 %	45,45 %	33,19 %	63,03 %	66,94 %
net 103	new_classoutput	11,06 %	5,45 %	3,79 %	11,83 %	8,09 %	4,29 %	15,34 %	6,64 %	7,56 %	10,74 %
	new_prob	11,06 %	5,45 %	3,79 %	11,83 %	8,09 %	4,29 %	15,34 %	6,64 %	7,56 %	10,74 %
	new_fully	22,11 %	17,09 %	12,62 %	29,57 %	28,48 %	10,86 %	22,73 %	11,50 %	26,05 %	31,41 %
	new_dropout	46,73 %	26,91 %	24,61 %	44,09 %	56,63 %	22,29 %	33,52 %	22,12 %	47,06 %	58,68 %
	inception_5b-5x5_reduce	51,26 %	28,73 %	23,34 %	51,08 %	69,58 %	26,00 %	36,93 %	23,01 %	53,78 %	60,33 %
	inception_5a-3x3	53,27 %	30,18 %	25,87 %	48,92 %	69,26 %	26,00 %	38,64 %	28,32 %	56,30 %	61,16 %
	inception_4c-5x5	49,25 %	26,91 %	23,03 %	43,01 %	66,67 %	26,00 %	33,52 %	24,34 %	49,58 %	59,50 %
	inception_4a-3x3_reduce	55,78 %	37,45 %	35,02 %	47,85 %	79,29 %	38,29 %	45,45 %	33,19 %	63,03 %	66,94 %
net 134	new_classoutput	9,05 %	6,55 %	4,42 %	13,44 %	11,00 %	4,29 %	10,80 %	5,75 %	10,92 %	10,74 %
	new_prob	9,05 %	6,55 %	4,42 %	13,44 %	11,00 %	4,29 %	10,80 %	5,75 %	10,92 %	10,74 %
	new_fully	21,11 %	13,82 %	7,26 %	27,42 %	23,30 %	10,57 %	18,75 %	12,39 %	25,21 %	35,54 %
	new_dropout	42,71 %	24,36 %	18,93 %	42,74 %	56,96 %	19,71 %	35,23 %	20,80 %	47,06 %	60,33 %
	inception_5b-5x5_reduce	45,73 %	27,64 %	21,77 %	46,24 %	61,17 %	21,14 %	38,07 %	21,24 %	52,10 %	63,64 %
	inception_5a-3x3	49,75 %	28,36 %	23,66 %	47,85 %	66,99 %	25,43 %	39,20 %	26,55 %	52,10 %	61,16 %
	inception_4c-5x5	49,25 %	26,91 %	23,03 %	43,01 %	66,67 %	26,00 %	33,52 %	24,34 %	49,58 %	59,50 %
	inception_4a-3x3_reduce	55,78 %	37,82 %	35,02 %	47,85 %	79,29 %	38,29 %	45,45 %	33,19 %	63,03 %	66,94 %

Tabla A.17: Tabla completa de aciertos por clase, red y capa.

B

Imágenes

B.1. Histogramas

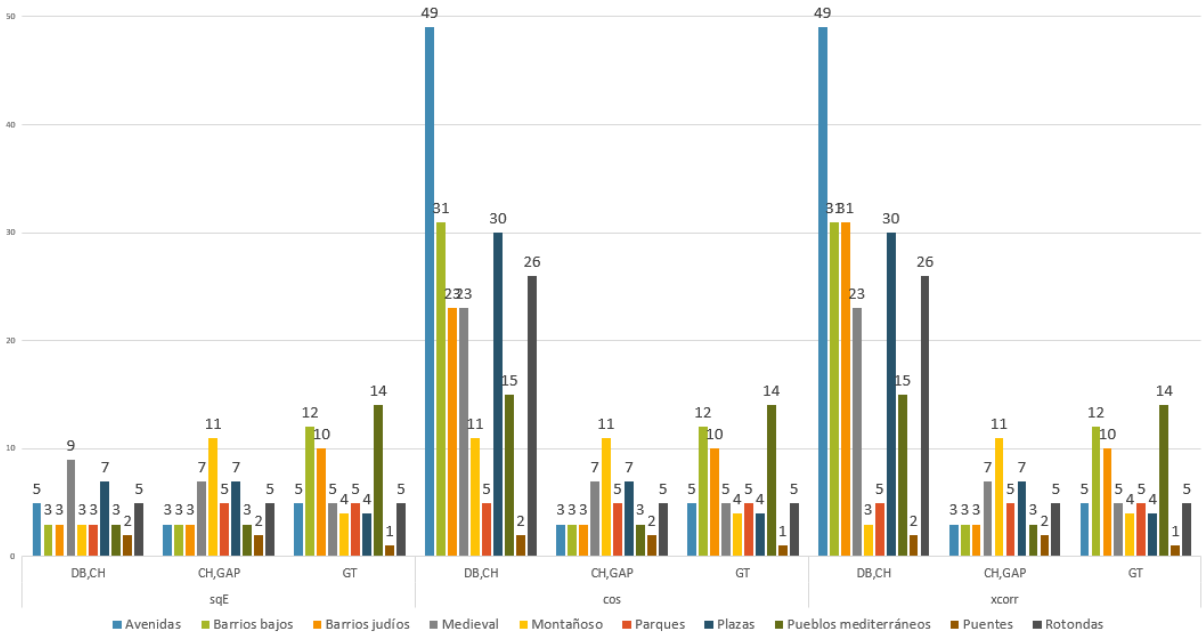


Figura B.1: Histograma resultado del *clustering* para todos los casos, más el *ground truth*, con el algoritmo Linkage.

B.2. Más ejemplos de agrupaciones obtenidas

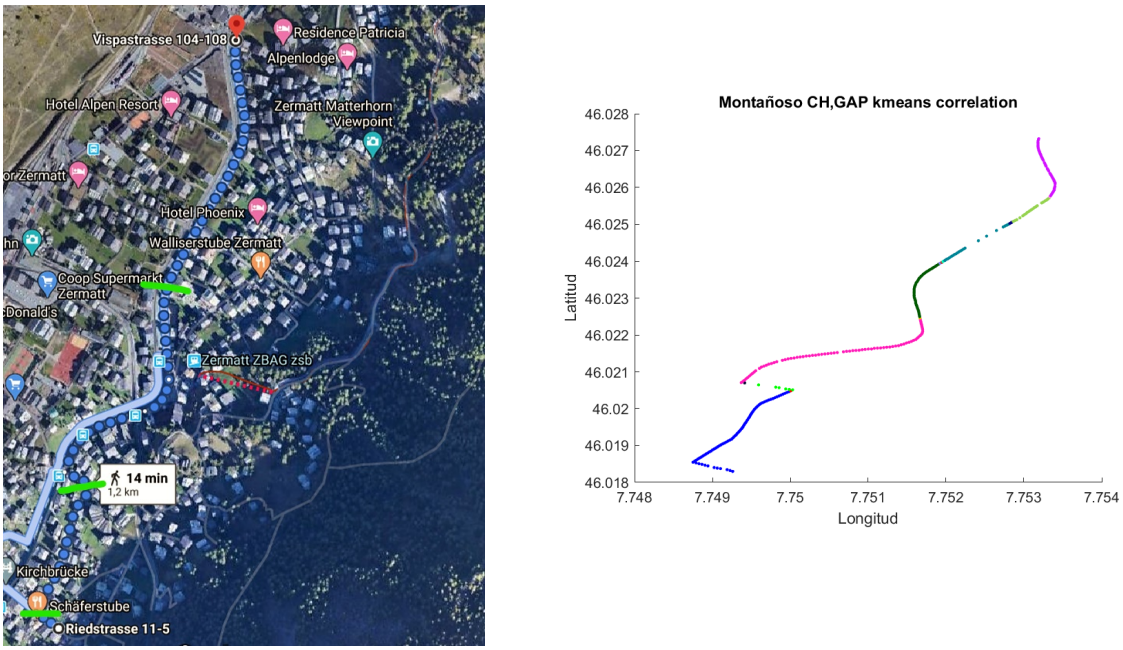


Figura B.2: Trayectoria clase Montañoso extraída de Google Street View y representación de los *clústers* anotados: 4 y obtenidos: 15.

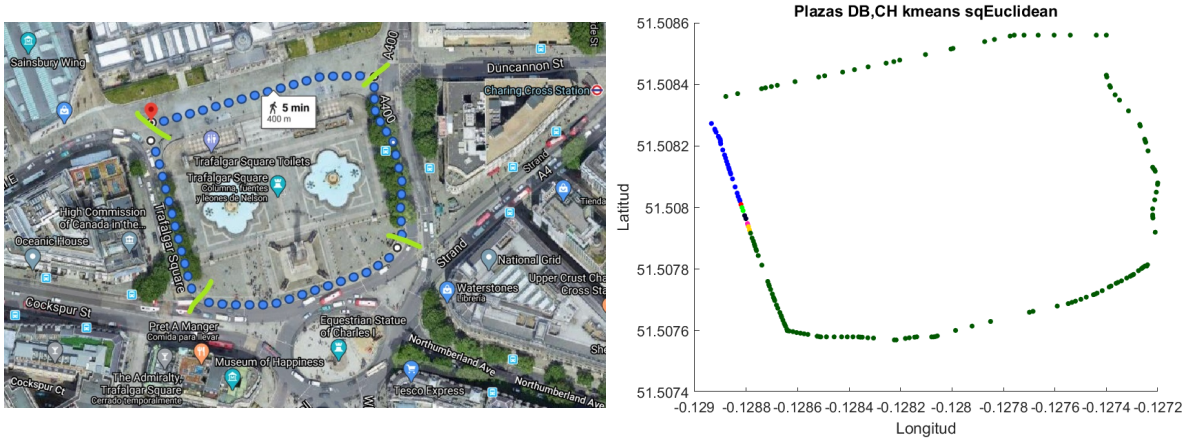


Figura B.3: Trayectoria clase Plazas extraída de Google Street View y representación de los *clústers* anotados: 4 y obtenidos: 7.

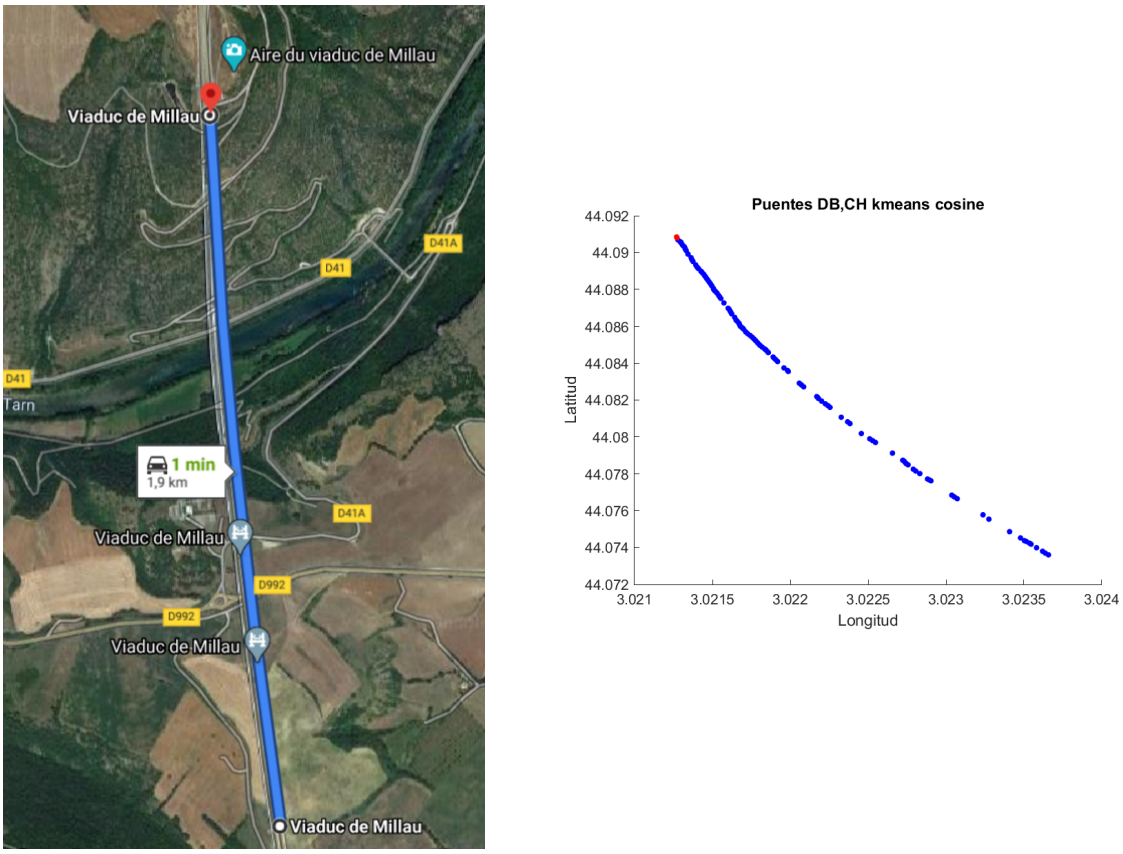


Figura B.4: Trayectoria clase Puentes extraída de Google Street View y representación de los *clústers* anotados: 1 y obtenidos: 2.